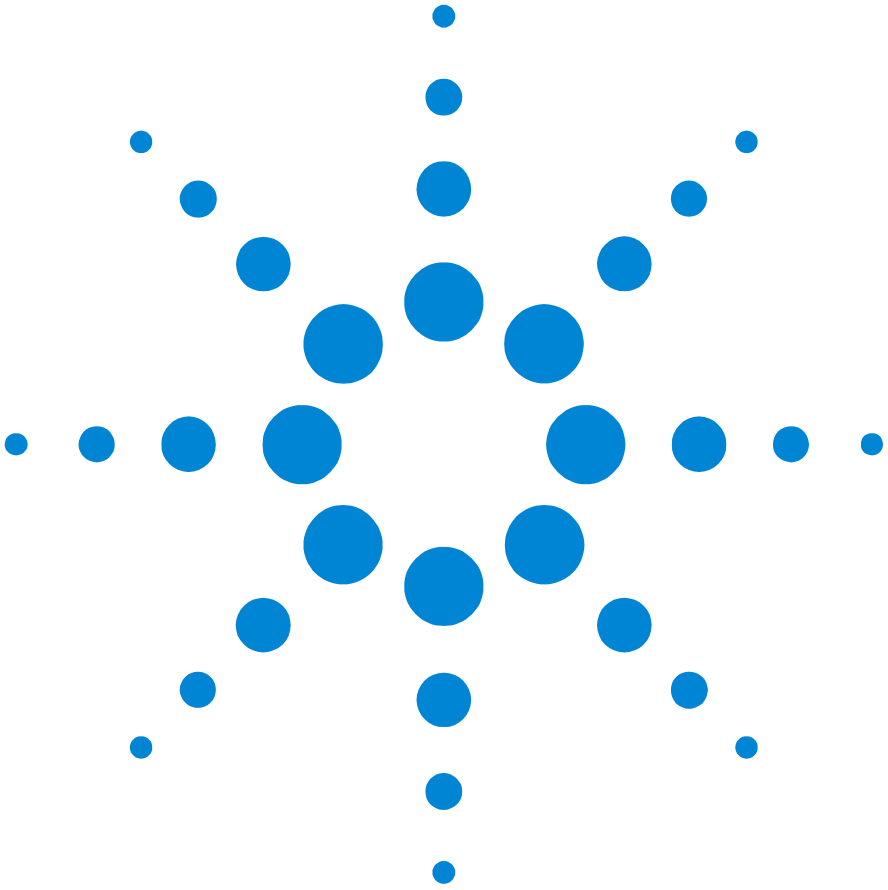


Agilent N4901 Serial BERT

Programming Guide

Agilent Technologies

Important Notice

Revision

Revision 2.0, May 2004

Printed in Germany

Agilent Technologies
Herrenberger Straße 130
D-71034 Böblingen
Germany

Authors: t3 medien GmbH

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING/DANGER

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Trademarks

Windows NT ® and MS Windows ® are U.S. registered trademarks of Microsoft Corporation.

Contents

Contents

Programming Basics

This document provides the information you need for programming the Agilent N4901A/2A Serial BERT using the Agilent VISA I/O libraries. Familiarity with the Agilent VISA I/O libraries is instrumental in understanding remote programming of the Serial BERT.

See the user documentation delivered with the Agilent VISA I/O libraries for information on how to use them.

CAUTION

The following pattern generator ports must be terminated with 50 Ω if they are not connected:

- Data Out
- $\overline{\text{Data Out}}$
- Clock Out
- $\overline{\text{Clock Out}}$

The following procedure is recommended when setting up a test:

1. If your DUT can handle 0 V, disable the outputs.

   IVI-COM: IAgilentN490xPGGlobal.OutputsEnable = False

   SCPI: OUTPut[1]:CENTer DISConnect

   The pattern generator's Data Out and Clock Out outputs are set to 0 V.

2. Connect the DUT as necessary.
3. Terminate any non-connected Data Out and Clock Out ports (normal and complementary).
4. If the outputs are disabled, reenable them.

   IVI-COM: IAgilentN490xPGGlobal.OutputsEnable = True

   SCPI: OUTPut[1]:CENTer CONNect

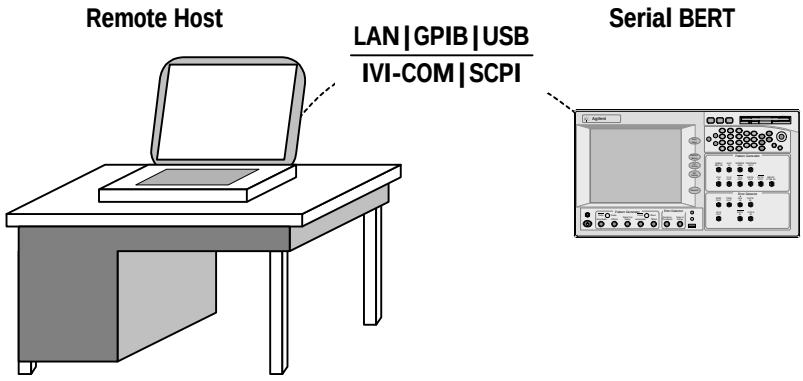
Before You Begin

This section provides background information that you need before you start with remote programming. It contains the following subjects:

Communication Overview

Communication with the Serial BERT is based on a host-client protocol. The server is the Serial BERT itself, the host is the remote client. The host requests the server to carry out specific actions; the Serial BERT carries out the actions and returns the results (if a query was sent).

Figure 1 Serial BERT Remote Communication



The Serial BERT uses either a SCPI interface or IVI-COM interface for for communicating with the outside world. See *"A Typical Serial BERT Program" on page 13* for information on getting started with remote programming for the Serial BERT.

The Serial BERT's advanced measurements can only be accessed over the LAN interface. See the *Measurement Software Programming Guide* for more information on programming the measurements.

Connecting to the Serial BERT

NOTE To communicate with the Serial BERT from a remote machine, the Agilent VISA I/O libraries must be installed on this machine.

The following descriptions only provide you with the information you need from the Serial BERT. For complete instructions on how to establish connections to the Serial BERT, refer to the user documentation delivered with the VISA I/O libraries.

The VISA I/O libraries offer the following possibilities for remotely connecting to and controlling the Serial BERT:

LAN The Serial BERT's network settings are managed by the operating system. You can use the **IPCONFIG** command in the command window to get the network settings.

The steps for setting up the network connection are OS-dependent (Serial BERT's OS is Windows XP). Contact your network administrator if you need help in defining the network settings.

GPIB To connect to the Serial BERT via GPIB, you have to have the Serial BERT's GPIB address.

The address is displayed on the user interface. The default address is 14. See the online Help for details on how to set the GPIB address.

NOTE When setting the GPIB address, it is recommended that you *do not* use the GPIB address 21. This address is reserved for GPIB controllers.

USB The Serial BERT has a USB port on the rear of the instrument that you can use to connect it to a PC. This is the non-flat USB port below the GPIB port.

To connect to the Serial BERT via USB, you need the Serial BERT's USB ID. You can either use the full VISA resource string or assign an alias. See the Agilent VISA I/O libraries documentation for details.

Instrument Behavior

The Serial BERT behaves as follows when it is turned on (or after a power-cycle):

Instrument Mode

At power on, the Serial BERT will return to the same mode as it was powered down. Normally, once it has booted, the Serial BERT is ready for either front panel operation or remote operation.

Registers and Filters

At power-on, the state of the registers and filters are:

- Normal operation

  The initial state of the registers and transition filters will be saved in the event of a power failure.

- Initial power-on

  All registers and filters are disabled except the PON, CME and EXE bits of the Standard Event Status Register and its summary bit in the Status Byte.

  The transition filters will be set to allow all conditions and events to pass.

The event registers and the error queue are cleared at each and every power-up.

Overheat Protection

The Serial BERT protects itself from damage by overheating by shutting itself down in such cases.

If the temperature of the pattern generator or error detector generator exceeds a certain threshold, the OVERHEAT bit in the Operation register is set.

There are two thresholds: caution and warning. These both set the same bit: you cannot programmatically get the threshold.

The caution threshold is not critical. You have enough time to save your current settings and gracefully shut down the instrument.

The warning threshold is critical. If this threshold is reached, the instrument will immediately shut itself down.

Overtemperature can be programmatically detected by querying the Status byte (*STB). In case of overheating by either the error detector or pattern generator, the Operation bit (bit 7) in the Status byte will high, as will the OVERHEAT bit in the Operation register. See *"How Serial BERT Uses Status Registers" on page 23* for details on reading the status registers.

You can identify whether the error detector or pattern generator is overheating by running a self-test on both devices. To run a self-test:

- IVI-COM: IIviDriverUtility.SelfTest
- SCPI: TEST:EXECute?

See also the Serial BERT User Guide (or online Help) for details.

Operation Modes

The Serial BERT can be operated in one of two modes: local or remote.

Local Mode

In *local* mode, all the front panel controls are responsive and control the instrument.

Remote Mode

In *remote* mode, the front panel controls are inoperative and the instrument is controlled by the client. The front panel display reflects the remote programming commands received.

The Serial BERT automatically enters remote mode when a command has been received from the client. This is indicated at the top of the front panel (the **RMT** status light).

Returning to Local Mode

To return to local mode, press the front panel **Local** key. When you power-cycle the instrument, it will also start in local mode.

A Typical Serial BERT Program

The Serial BERT can be controlled by a remote program using the IVI-COM driver.

The sections of this Help provide you with information you can use to quickly get started with your first program. The examples here are written for Visual Basic 6.0, but can also be ported to any programming language supported by IVI-COM.

See the Serial BERT driver's Help for more information.

Prerequisites

Before you can control a Serial BERT remotely, the client computer (your PC, the Serial BERT is the host) must meet the following prerequisites:

- Agilent VISA I/O libraries installed
- IVI-COM driver installed
- Configured IO connection to the Serial BERT (you should be able to find the Serial BERT with the I/O libraries VISA assistant)

Initializing the Connection to the Serial BERT

The first step in setting up a program for controlling the Serial BERT is to create an object that corresponds to the instrument. You can either use the Serial BERT class (AgilentN490x), or you can use the IVI-compliant Agilent BERT class (AgilentBert).

TIP If you set up your code using the AgilentBert class, you can easily port your programs to other IVI-compliant Agilent instruments. As Agilent's fulfillment of the IVI-compliance requirements, this class is shared by all other Agilent IVI-compliant instruments.

The examples in this document show how to set up a reference to the AgilentBert class and use this class.

Initializing the Connection – Procedures

The following code shows you how you would establish the connection to the instrument. The ResourceName ("TCPIP1::10.0.0.207::inst0::INSTR") must be replaced by the instrument's address string from the VISA Assistant.

```
' First our declarations...
Private myN490X As AgilentN490x
Private myBERT As AgilentBert
Private myPG As AgilentBertLib.IAgilentBertPG
Private myPGClock As AgilentBertLib.IAgilentBertPGClock
Private myPGOut As AgilentBertLib.IAgilentBertPGOutput
Private myEDDataIn As AgilentBertLib.IAgilentBertEDDataIn

Private Sub Form_Load()

Set myN490X = New AgilentN490x
Set myBERT = myN490x.IAgilentBert

myBERT.Initialize "TCPIP1::10.0.0.207::inst0::INSTR", True, True

End Sub

Private Sub Form_Unload(Cancel As Integer)
myBERT.Close
End Sub
```

Working with the IVI-COM Objects

The Serial BERT IVI-COM driver uses a hierarchical class structure that follows the build up of the instrument. For example, the instrument itself is represented by the class AgilentN490x. The pattern generator is represented by the class IAgilentN490xPG.

To access the instrument's pattern generator, you have to first access the object, then the object's pattern generator collection, and finally the actual pattern generator.

The items in the collections are accessed by the name. The easiest way to get the name (if you do not know it) is through the collection's Name property.

Working with the IVI-COM Objects – Procedures

The following example shows you how to set up different objects for controlling the Serial BERT. These objects are used in the following examples.

```
Private Sub InitializeObjects()
' TIP: Call this sub from the Form_Load sub.
Dim EDName as String
With myBERT
   ' Get the pattern generator using the naming conventions
   Set myPG = .PGs.Item("PG1")
   ' Use the myPG object to get sub-items
   Set myPGClock = myPG.Clock
   Set myPGOut = myPG.Outputs.Item("PGOutput1")

   ' Get the error detector by catching and using its name:
   EDName =.EDs.Name(1)
   Set myED = .EDs.Item(EDName)
   Set myEDDataIn = myED.Input.DataIns.Item("EDDataIn1")


End With

End Sub
```

Changing Instrument Parameters

The following examples show you how to:

- Change the pattern generator's clock rate and voltages
- Trigger auto-synchronization
- Set up a pattern

Change the Pattern Generator's Clock Rate and Output Voltages

The following code sets the pattern generator's clock frequency and toggles as example the offset voltage between 0 and 0.5 Volts.

```
Private Sub SetUpPG
' Set the clock frequency
myPGClock.Frequency = 3 * 10 ^ 9

' Toggle the offset voltage (for demo purposes)
If myPGOut.OutVoltage.VOffset = 0 Then
   myPGOut.OutVoltage.VOffset = 0.5
Else
   myPGOut.OutVoltage.VOffset = 0
End If
End Sub
```

Trigger Synchronization

The following code causes the error detector to synchronize.

```
Private Sub RunSync()
' First run the synchronization
myEDDataIn.Sampling.AutoAlign

' And then align the data
myEDDataIn.Synchronisation.SyncNow

End Sub
```

Set Up a Pattern

The following code shows you how to set up a pattern. It additionally shows a small function that converts strings into arrays that Visual Basic can handle.

```
Private Sub SetUpPattern()
Dim myPattern As AgilentBertLib.IAgilentBertLocalPatternfile

' Use local pattern 13 to save the pattern files
' to a different location

Set myPattern = myBERT.LocalPatternfiles._
         Item(myBERT.LocalPatternfiles.Name(13))

With myPattern
   ' Set the length and description
   .Length = 8
   .Description = "Test pattern"

   ' Define the pattern to be alternate, set the data
   ' For VB, we have to convert the data to an array of doubles
   ' See function below for details
   .Alternate = True
   .SetData 1, AgilentLocalPatternFormatBin, _
      SetPatternData("00001111", AgilentLocalPatternFormatBin)
   .SetData 2, AgilentLocalPatternFormatBin,
      SetPatternData("11111111", AgilentLocalPatternFormatBin)

End With
' And now load the pattern to the pattern generator
myPGOut.SelectData AgilentN490xPGOutputSelectFile, _
      myPattern.Location

' And to the error detector
myEDDataIn.SelectData AgilentBertEDDataInSelectFile, _
      myPattern.Location


End Sub

Private Function SetPatternData(DataString As String, _
   Format As AgilentBertLib.AgilentBertEDPatternFormatEnum)

Dim myPattern() As Double
Dim ix As Integer
ReDim myPattern(Len(DataString) - 1)

For ix = 1 To Len(DataString)
   Select Case Format
   ' How to interpret the string depends on the format
      Case AgilentBertEDPatternFormatBin
         myPattern(ix - 1) = CByte(Mid(DataString, ix, 1))
      Case AgilentBertEDPatternFormatHex
         myPattern(ix - 1) = CByte("&H" & Mid(DataString, ix, 1))
```

```
      Case AgilentBertEDPatternFormatRaw
         myPattern(ix - 1) = CByte(Mid(DataString, ix, 1))
   End Select
Next
SetPatternData = myPattern
End Function
```

Recommended Programming Techniques

This chapter provides some recommended techniques you should use when programming the Serial BERT.

Controlling the Output Levels

How Serial BERT Controls the Output Levels

When the output levels are changed at the Serial BERT's data and clock output ports, four parameters are changed:

- V_{hi}
- V_{lo}
- V_{ampt}
- V_{offs}

The Serial BERT groups these parameters into "pairs" (V_{ampt}/V_{offs}, V_{hi}/V_{lo}). If one of these values is modified, its "partner" remains constant, and the values in the other pair are modified accordingly. For example, if V_{ampt} is changed, V_{offs} stays constant, and V_{hi} and V_{lo} are modified accordingly.

Changing the Voltages with IVI-COM

The IVI-COM driver provides a convenient function for setting V_{ampt} and V_{offs} : `Configure`. To set the pattern generator's data output voltage:

```
Private Sub SetPGDataOutVolt()
Dim myPG As AgilentN490xLib.IAgilentN490xPG
Dim myPGOut As AgilentN490xLib.IAgilentN490xPGOutput
```

```
Set myPG = myBERT.PGs.Item("PG1")
Set myPGOut = myPG.Outputs.Item("PGOutput1")

myPGOut.OutVoltage.Configure 1.5, 0.5, _
         myPGOut.OutVoltage.VTermination
End Sub
```

Changing the Voltages with SCPI

The following command shows how you would set the data output so that it has an amplitude of 1.5 V and an offset of 0.5 V:

```
SOUR:VOLT:AMPT 1.5; OFFS 0.5
```

This sets the output accordingly (V_{Hi} = 1.25 V, V_{Lo} = –0.25).

Conversely, you could set V_{Hi} and V_{Lo} directly:

```
SOUR:VOLT:HIGH 1.25; LOW -0.25
```

This has the same effect.

Allowing Serial BERT to Settle

When patterns are sent to the pattern generator or error detector, the Serial BERT requires some time to settle before. The following topics explain how the instruments react to pattern changes.

How Pattern Changes Affect the Pattern Generator

The Serial BERT requires some time to change the patterns at the pattern generator and error detector. This is particularly true for large text-based (ASCII) patterns that have to be loaded from the file system. In such a case, it is a recommended technique to always query the Serial BERT's Operation Complete status after changing the pattern.

How Pattern Changes Affect the Error Detector

When the pattern changes, the error detector has to resync to the new incoming signal. Depending on the signal, the alignment method used, and the desired BER threshold, this procedure can take up to half a minute or more.

Determining if Conditions have Settled

When patterns have been changed, you should check the status registers to make sure that the operation is complete before continuing with your program.

Checking the Settling with IVI-COM

The following example illustrates how to check the synchronization status using IVI-COM.

```
Private Sub CheckSyncStatus()
Dim BERTStatus As AgilentN490xLib.IAgilentN490xStatus
Dim myED As AgilentN490xLib.IAgilentN490xED
Dim myPG As AgilentN490xLib.IAgilentN490xPG

Set BERTStatus = myBERT.Status
Set myED = myBERT.EDs.Item("ED1")
Set myPG = myBERT.PGs.Item("PG1")

' First enable the register of interest:
' Operation register, bit 10, positive transition
BERTStatus.Register(AgilentN490xStatusRegisterOperation, _
      AgilentN490xStatusSubRegisterEnable) = &H400
BERTStatus.Register(AgilentN490xStatusRegisterOperation, _
      AgilentN490xStatusSubRegisterPositiveTransition) = &H400
BERTStatus.Register(AgilentN490xStatusRegisterOperation, _
      AgilentN490xStatusSubRegisterNegativeTransition) = &H400

' Standard Event register, bit 0, positive transition
BERTStatus.Register(AgilentN490xStatusRegisterStandardEvent, _
      AgilentN490xStatusSubRegisterEnable) = 1

' Now clear the registers
BERTStatus.Clear

' ED should track the PG
myED.Input.DataIns.Item("EDDataIn1").TrackingEnabled = True

' Load the pattern
myPG.Outputs.Item("PGOutput1").SelectData
AgilentN490xPGOutputSelectFile, "testptr.ptrn"

' Just wait until the Operation bit goes low
Do While BERTStatus.SerialPoll And &H80
   DoEvents
Loop
End Sub
```

Checking the Settling with SCPI

The following example illustrates how to check the synchronization status using SCPI.

```
/* We need to check sync loss bit
   of the Questionable register (bit 10) */

const unsigned int QUESTION_REG_10 = 2^10;

unsigned int question_reg;
unsigned int opc_stat;

/* Make sure the error detector tracks
   the pattern generator */
viPrintf(vi, "SENSe1:PATTern:TRACk ON\n");

/* Load a pattern file to the instruments */
viPrintf(vi, "SOURce1:PATTern:SELect FILename, testfile.ptrn\n");

/* Wait until the instrument is in operational state */
viQueryf(vi, "*OPC?\n", "%d", &opc_stat);

do
{
   /* Get the Questionable register */
   viQueryf (vi, "STATus:QUEStionable:CONDition?\n", "%d",
            &question_reg);

   /* Loop until the sync loss bit goes low */
}
while(question_reg && QUESTION_REG_10);
```

If Question_con_reg returns a value that includes bit 10 (“1024”), this is an indication that the error detector has not yet synchronized to the new pattern. In this case, the instrument has not yet settled.

NOTE File accessing, especially for large files can take some time. Control programs must be prepared for time-outs of this size.

Reading the Serial BERT's Status

The Serial BERT has a set of status registers that you can use to monitor the status of the hardware, software, and any running tests.

Overview of Registers

Specifically, it has the following registers:

- Status Byte

  The Status Byte is a single register that stores the events occurring on the other registers.

- Standard Event Status Register

  The Standard Event Status Register monitors some non-critical errors and basic operations.

- Questionable Data Status Register

  The bits in the Questionable Data Status Register are set when certain events occur in the Serial BERT that can lead to questionable results.

- Operation Status Register

  The Operation Status Register indicates when certain operations have been completed.

- Clock Loss Status Register

  The Clock Loss Status Register indicates if either the Serial BERT's pattern generator or error detector have lost the clock signal.

How Serial BERT Uses Status Registers

You can determine the state of certain instrument hardware and firmware events and conditions by programming the status register system.

The following subsections provide you with details about the Serial BERT's status system.

Overview of the Serial BERT's Status System

The Serial BERT has status reporting features that give important information about events and conditions within the instrument. For example, a flag may be set to indicate the end of a measurement or perhaps a command error. To access this information, it is necessary to query a set of registers using SCPI.

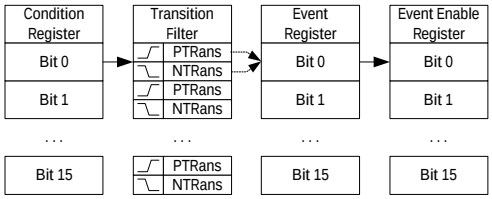
Serial BERT's Status System Structure

The Serial BERT's status system is comprised of multiple registers that are arranged in a hierarchical order. The lower-level status registers propagate their data to the higher-level registers in the data structures by means of summary bits. The Status Byte register is at the top of the hierarchy and contains general status information for the Serial BERT's events and conditions. All other individual registers are used to determine the specific events or conditions.

For figures showing Serial BERT's status registers, see *"Serial BERT Register Model" on page 25*.

Status Register Group Model

The following figure illustrates the typical structure of a status register.



As shown in this figure, most status registers actually consist of five registers:

- Condition

  The condition register continuously monitors the hardware and firmware status of the instrument. There is no latching or buffering for a condition register. It is updated in real time.

  This register is read by the CONDition? SCPI commands.

- Negative Transition

  The negative transition register specifies the bits in the condition register that will set corresponding bits in the event register when the condition bit changes from 1 to 0.

  This register is set and read by the NTRAnsition[?] SCPI commands.

- Positive Transition

  A positive transition register specifies the bits in the condition register that will set corresponding bits in the event register when the condition bit changes from 0 to 1.

- Event

  An event register latches transition events from the condition register as specified by the positive and negative transition filters. Bits in the event register are latched, and once set, they remain set until cleared by either querying the register contents or sending the *CLS command.

- Event Enable

  An enable register specifies the bits in the event register that can generate a summary bit. Summary bits are, in turn, used by the next higher register.

The registers work together as follows:

1 The *Condition Register* corresponds to a condition on the hardware or in the software. If the monitored condition is present, the corresponding bit is high.

2 The *Transition Registers* monitor changes in the Condition Register. If the Positive Transition Register is configured to watch for a condition, when this condition occurs, and the bit in the Condition Register goes high, the Positive Transition Register passes this event to the *Event Register*.

3 If this bit is enabled in the *Enable Event Register*, a summary bit is generated in the next higher register. For the higher register, this is the Condition Register, and the event is handled the same way as described here.

NOTE The transition and enable registers for the Failure Status register (and its subregisters) cannot be modified.

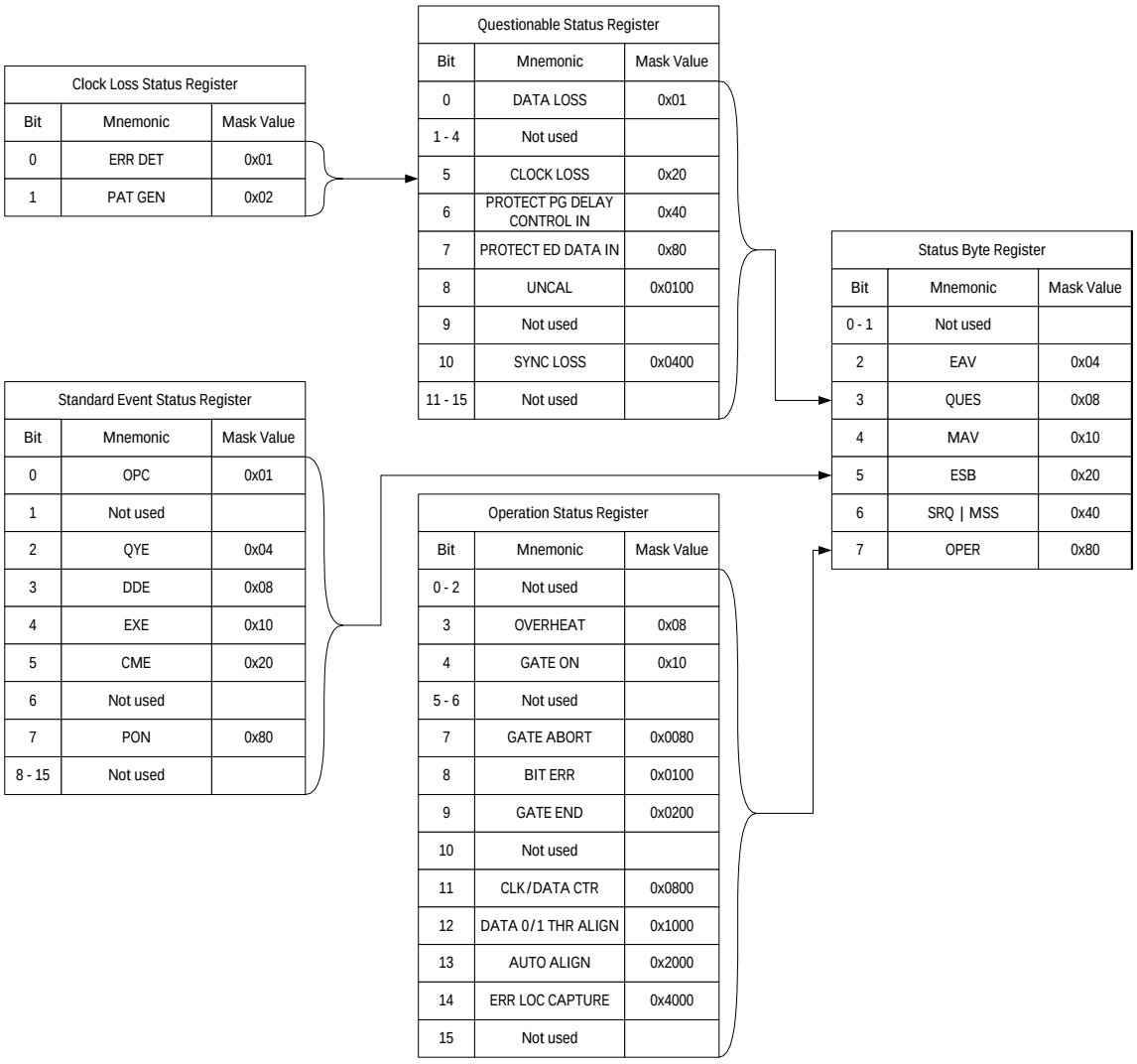
See *"Preparing the Registers (IVI-COM)" on page 35* for an example of how to access the registers in IVI-COM. See *"Preparing the Registers (SCPI)" on page 37* for a SCPI example.

Serial BERT Register Model

The Serial BERT has the following status registers:

- Status Byte
- Standard Event Status
- Questionable Data Status
- Operation Status
- Clock Loss

The following figure shows the Serial BERT's status register hierarchy. This figure shows the Status Byte.



Clock Loss Status Register

Bit	Mnemonic	Mask Value
0	ERR DET	0x01
1	PAT GEN	0x02

Questionable Status Register

Bit	Mnemonic	Mask Value
0	DATA LOSS	0x01
1 - 4	Not used	
5	CLOCK LOSS	0x20
6	PROTECT PG DELAY CONTROL IN	0x40
7	PROTECT ED DATA IN	0x80
8	UNCAL	0x0100
9	Not used	
10	SYNC LOSS	0x0400
11 - 15	Not used	

Status Byte Register

Bit	Mnemonic	Mask Value
0 - 1	Not used	
2	EAV	0x04
3	QUES	0x08
4	MAV	0x10
5	ESB	0x20
6	SRQ | MSS	0x40
7	OPER	0x80

Standard Event Status Register

Bit	Mnemonic	Mask Value
0	OPC	0x01
1	Not used	
2	QYE	0x04
3	DDE	0x08
4	EXE	0x10
5	CME	0x20
6	Not used	
7	PON	0x80
8 - 15	Not used	

Operation Status Register

Bit	Mnemonic	Mask Value
0 - 2	Not used	
3	OVERHEAT	0x08
4	GATE ON	0x10
5 - 6	Not used	
7	GATE ABORT	0x0080
8	BIT ERR	0x0100
9	GATE END	0x0200
10	Not used	
11	CLK/DATA CTR	0x0800
12	DATA 0/1 THR ALIGN	0x1000
13	AUTO ALIGN	0x2000
14	ERR LOC CAPTURE	0x4000
15	Not used	

Status Byte

The Status Byte is the summary register to which the other registers report. Each reporting register is assigned a bit in the Status Byte Register.

The bits in the Status Byte byte have the following meaning:

Bit	Mnemonic	Description
0	Not used	
1	Not used	
2	EAV	Error available: The error queue contains at least one message.
3	QUES	A bit has been set in the Questionable Data Status register (indicates that a signal is of questionable quality).
4	MAV	Message available: There is at least one message in the message queue.
5	ESB	A bit in the Standard Event Register has been set.
6	SRQ or MSS	Value depends on the polling method; see below for details.
7	OPER	A bit in the Operation Status Register has been set.

Bit 6 Bit 6 has two definitions, depending on how the access is polled:

- Serial Poll

  If the value of the register is read using the serial poll (SPOLL), bit 6 is referred to as the Service Request (SRQ) Bit. It is used to interrupt and inform the active controller that the instrument has set the service request control line, SRQ.

- *STB?

  If the register is read using the common command *STB? , bit 6 is referred to as the master summary bit or MSS bit. This bit indicates that the instrument has requested service. The MSS bit is not cleared when the register is read using the *STB? command. It always reflects the current status of all the instrument's status registers.

Standard Event Status Register

The Standard Event Status register is a 16-bit register group that gives general-purpose information about the instrument. It sets bit 5 in the Status Byte.

Bit	Mnemonic	Description
0	OPC	Operation Complete bit. It is set in response to the *OPC command, but only if the instrument has completed all its pending operations.
1	Not used	
2	QYE	Query error bit. It indicates that there is a problem with the output data queue. There has been an attempt to read the queue when it is empty, the output data has been lost, or the query command has been interrupted.
3	DDE	Device-dependent error bit. It is set when an instrument-specific error has occurred.
4	EXE	Execution error bit. It is set when a command (GPIB instrument specific) cannot be executed due to an out of range parameter or some instrument condition that prevents execution.
5	CME	Command error bit. It is set whenever the instrument detects an error in the format or content of the program message (usually a bad header, missing argument, or wrong data type etc.).
6	Not used	
7	PON	Power-on bit. It is set each time the instrument is powered from off to on.
8-15	Not used	

NOTE This register is compatible with the generalized status register model. It is comprised of an event and enable register, but no condition register or transition filter. All positive transitions in this register are latched.

Clock Loss Register

The Clock Loss Register group indicates whether the pattern generator or error detector has experienced a clock signal loss. The output of this register sets bits 5 and 9 (Clock Loss) in the Questionable Status Register.

Bit	Mnemonic	Description
0	ERR DET	Clock loss condition at the error detector.
1	PAT GEN	Clock loss condition at the pattern generator.
2-15	Not used	

Questionable Status Register

The Questionable Status Register group indicates that a currently running measurement is of questionable quality. The output of this register sets bit 3 of the Status Byte.

Bit	Mnemonic	Description
0	DATA LOSS	This bit is set when the data source is turned off, not connected, or the cables or device is faulty. This bit can also set when the 0/1 threshold is not in the eye limits of the incoming data signal. In this last case, use Auto Align or select Avg 0/1 Threshold.
1-4	Not used	
5	CLOCK LOSS	This bit is set when the pattern generator receives no external clock signal or the error detector receives no clock input signal. To find out which of the 2 events is causing this bit to set, you must poll the Clock Loss Status Register; see *"Clock Loss Register" on page 28*.
6	PROTECTED DATA IN	This bit indicates that the protection mechanism for the Data Input port of the error detector was activated, e.g. the voltage or current measured at this port was out of range.
7	PROTECT PG DELAY CTRL IN	This bit indicates that the protection mechanism for the Delay Control Input port of the pattern generator was activated, e.g. the voltage or current measured at this port was out of range.
8	UNCAL	This bit is set when the serial number of the installed pattern generator or error detector tray does not match the calibration file in the instrument.
9	Not used	
10	SYNC LOSS	This bit is set when the error detector pattern does not match the incoming data pattern or the BER of your device is higher than the sync threshold.
11-15	Not used	

Operation Status Register

The output of this register gives information about the current operation the instrument is performing. It sets bit 7 of the Status Byte.

Bit	Mnemonic	Description
0-2	Not used	
3	OVERHEAT	Either the pattern generator or error detector has a higher-than-normal temperature.
4	GATE ON	An accumulated measurement is in progress.
5-6	Not used	
7	GATE ABORT	Indicates that the repetitive accumulation period was aborted.
8	BIT ERR	The instrument has detected a bit error.
9-10	Not used	
11	CLK/DATA CTR	Indicates that the clock/data alignment is in progress.
12	DATA 0/1 THR ALIGN	Indicates that the 0/1 threshold alignment is in progress.
13	AUTO ALIGN	Indicates that the auto alignment is in progress.
14	ERR LOC CAPTURE	Indicates that there is an Error Location Capture measurement in progress.
15	Not used	

Using Error Location Capture

What is Error Location Capture?

Error Location Capture allows to capture the position of an errored bit in the incoming bitstream. The instrument searches for the first bit errored in the incoming bitstream. The address of the errored bit is saved after the error is located.

This feature can be used to find rare or random errors. A DUT could have problems handling long series of zeroes. Error Location Capture can be used to locate the bit errors in such cases.

Restrictions for Error Location Capture

Error Location Capture is subject to the following restrictions:

- Only memory-based patterns with a unique 48-bit pattern (detect word) are allowed.
- The error detector must be aligned to the incoming stream.
- No alignment features can run during error location capture: Auto Align, 0/1 Threshold Center, Data Center
- No other advanced measurement (Output Timing, Output Levels, etc.) can be running.
- Error Location Capture can only run when the BER Location Mode is set to more than one bit (for example, all bits, or a block with a length > 1).

How to Run Error Location Capture

The following steps are recommended for running Error Location Capture:

1. Clear the Status registers.
2. Set up the status registers to ensure that the ERR LOC CAPTURE bit is monitored.

   The Operation Status register should catch positive transitions on the ERR LOC CAPTURE bit (bit 14).
3. Set the Error Location mode to scan the incoming bit stream for all bits.
4. Start the Error Location Capture. This is an overlapped command.

5. Set up a loop that queries the Operation register until the Error Location bit goes high. This indicates that the Error Location Capture has started.
6. Set up a loop that queries the Error Location status to see if it is still running.
7. When Error Location Capture has quit, check the state.
8. If the state is valid, query the location of the errored bit.

NOTE Because Error Location Capture would run forever if no errors are detected, it is recommended to also set up a time-out in your program.

How to Abort Error Location Capture

Error Location Capture runs until it detects an error and stops. If there are no errors in the data stream, it would run forever. It can also be interrupted by a remote program (or user) by the following actions:

- Disabling error location
  - IVI-COM: `IAgilentN490xEDErrorLocation.Mode = AgilentN490xEDErrorLocationModeOff`
  - SCPI: `SENS:ELOC OFF`
- Incorrect sample point (faulty measurement)
- Action not compatible with error location currently being performed, for example:
  - Selecting a new pattern
  - Changing the current pattern
  - Starting synchronization or alignment

There are various other actions that also abort the run. Changing the sampling point has no effect (as long as the sampling point does not leave the eye).

How to See if Error Location Capture is Running

To see if Error Location Capture is currently running:

1. Clear the Status system.
2. Call *OPC?. This shows that an overlapped command is running. It does not identify which overlapped command.
3. Query the Status register. If the Operation Status bit is high (bit 7), continue. Otherwise, Error Location Capture is not running.

4. Check the status of the Operation Status register. If the ERR LOC CAPTURE bit (bit 14) is high, Error Location Capture is running.
5. Check the status of Error Location Capture:

   IVI-COM: Call `IAgilentN490xEDErrorLocation.IsCaptureErrorsComplete`.

   SCPI: Call `SENS:ELOC?`. A 1 indicates that Error Location Capture has been triggered.
6. The SCPI command `SENS:ELOC:VERB?` may also indicate that Error Location Capture is running.

Understanding the Status

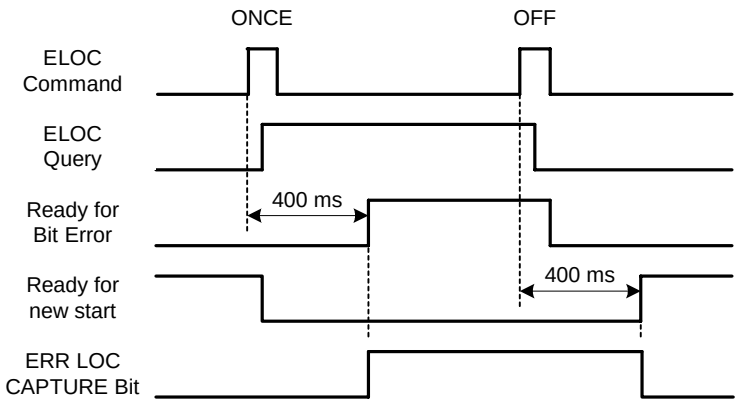
Error Location Capture is not immediately able to detect bit errors when the start command is given, and does not immediately terminate when the stop command is given. There is a specific delay that must be heeded. For remote programming, it is sufficient to wait 400 ms. It is also possible to check the status in the status registers.

The Error Location Capture bit (bit 14) in the Operation register indicates the true Error Location Capture status. If this bit is high, an Error Location Capture measurement is running.

The Error Location Capture query on the other hand returns the status of the last ELOC command (and not the true Error Location Capture status).

In other words, if the Error LocationLocation Capture query indicates that Error Location Capture is running, this actually means that Error Location Capture has been *triggered* (but is not necessarily running). Otherwise, it indicates that Error Location Capture has been either aborted (and may be still running) or successfully finished.

The following diagram indicates the value of the Error Location Capture bit in the Operation register and the Error Location Capture query return value:



Handling the Results

Once an Error Location Capture has been successfully finished, you can get the following results:

- Number of errored bits found
- Location of first errored bit
- Comparison pattern between expected pattern and received data

  The captured data is saved as an alternating pattern. Pattern A contains the expected data. Pattern B contains the errored data: 0s if the expected bits were also received, 1s for errored bits. To calculate the captured pattern, EXOR the bits from pattern A with the bits from pattern B.

  The pattern description contains the first error, the error count, date and time.

  The name of the pattern file is ELOC_RESULT_CURRENT.ptrn for the current capture and ELOC_RESULT_PREVIOUS.ptrn for the previous capture. These patterns are saved under C:\N4901A\Pattern on the machine with the firmware server.

How to Handle Run Errors

Errors in Error Location Capture are handled differently than standard instrument errors:

- Errors caused by starting or stopping Error Location Capture are put in the standard error queue.
- Internal run errors caused during Error Location Capture are neither put into the standard error queue nor reported by the status register's error flag. In such a case, the response to SENS:ELOC:VERB? is ELOC__FAILED.

Using Error Location Capture – Procedures

There is a slight delay (~400 ms) after error location capture is triggered before the measurement actually starts. There are two ways to handle this:

- For simplicity, just wait in your program 400 ms before continuing.
- For precision, monitor the error location status bit until it registers that error location has started.

The following code examples show how to set the registers and then run Error Location Capture.

Running ELOC in IVI-COM

The following subroutines show you how you can prepare the registers and then run error location.

Preparing the Registers (IVI-COM)

```
Private Sub PrepareRegisters()

' Define the classes;
' myBERT is the already created SerialBERT object
Dim myStatus As AgilentN490xLib.IAgilentN490xStatus
Dim myED As AgilentN490xLib.IAgilentN490xED
Dim myEDDataIn As AgilentN490xLib.IAgilentN490xEDDataIn
Dim myELOC As AgilentN490xLib.IAgilentN490xEDErrorLocation

Set myED = myBERT.EDs.Item("ED1")
Set myEDDataIn = myED.Input.DataIns.Item("EDDataIn1")
Set myELOC = myEDDataIn.ErrorLocation
Set myStatus = myBERT.Status

' Disable error location
myELOC.Mode = AgilentN490xEDErrorLocationModeOff
```

```
' Set the registers: &H4000 is bit 14, the ELOC bit
' in the Operation register

With myStatus
   .Register(AgilentN490xStatusRegisterOperation, _
      AgilentN490xStatusSubRegisterEnable) = &H4000
   .Register(AgilentN490xStatusRegisterOperation, _
      AgilentN490xStatusSubRegisterPositiveTransition) = &H4000
   .Register(AgilentN490xStatusRegisterOperation, _
      AgilentN490xStatusSubRegisterNegativeTransition) = 0
' And clear the registers
   .Clear
End With

End Sub
```

Running Error Location Capture

```
Private Sub RunErrorLocation()

' Define the classes;
' myBERT is the already created SerialBERT object
Dim myED As AgilentN490xLib.IAgilentN490xED
Dim myEDDataIn As AgilentN490xLib.IAgilentN490xEDDataIn
Dim myELOC As AgilentN490xLib.IAgilentN490xEDErrorLocation
Dim myStatus As AgilentN490xLib.IAgilentN490xStatus
Dim Timeout As Double
Dim bTimedOut As Boolean

Set myED = myBERT.EDs.Item("ED1")
Set myEDDataIn = myED.Input.DataIns.Item("EDDataIn1")
Set myELOC = myEDDataIn.ErrorLocation
Set myStatus = myBERT.Status

' Error location only runs for all bits
myELOC.Mode = AgilentN490xEDErrorLocationModeAllBits

' Start the actual capture
myELOC.CaptureErrors

' Wait until the ELOC bit goes high
Do Until myStatus.Register(AgilentN490xStatusRegisterOperation, _
   AgilentN490xStatusSubRegisterCondition) And &H4000
   DoEvents
Loop

' Wait until ELOC is finished
Do Until myELOC.IsCaptureErrorsComplete
   DoEvents
Loop
```

```
Select Case myELOC.ReadState
   Case AgilentN490xEDErrorLocationStateSuccess
   ' Get the error count and location
      Msgbox myELOC.ReadCount " errors found." & vbCrLF & _
         "First error found at: " & myELOC.BitAddress

   Case AgilentN490xEDErrorLocationStateFailed
      Msgbox "Error Location Capture failed."

   Case AgilentN490xEDErrorLocationStateAborted
      Msgbox "Error Location Capture aborted."
End Select

End Sub
```

Running ELOC in SCPI

Preparing the Registers (SCPI)

To verify that the instrument is ready by monitoring the error location status bit:

1 Disable error location capture:

```
SENSe1:ELOCation OFF
```

2 Activate the error location bit (bit 14) in the Operation register:

```
STATus:OPERation:ENABle 16384
```

3 Enable monitoring of positive transitions at the error location bit:

```
STATus:OPERation:PTRansition 16384
```

4 Disable monitoring of negative transitions at the error location bit:

```
STATus:OPERation:NTRansition 0
```

5 Clear the Operation register:

```
STATus:OPERation?
```

6 Check the status byte:

```
*STB?
```

The status byte should return 0.

Running Error Location Capture

To run error location capture in SCPI:

1 Start error location capture:

```
SENSe1:ELOCation ONCE
```

2 Set up a loop in your program and wait until the status byte value changes:

```
*STB?
```

This should return 0. Your loop should continue until the return value includes the Operation bit (7) (`*STB? && 2^7 > 0`).

3 Check the Operation status register:

```
STATus:OPERation?
```

When the error location has started, the return value should include the error location bit (`*STAT:OPER? && 2^14 > 0`).

4 Set up a loop in your program and wait until the error location capture has finished:

```
SENSe1:ELOCation?
```

This returns a 1 when error location capture has finished.

5 Verify that error location capture has valid results:

```
SENSe1:ELOCation:VERBose?
```

6 If `ELOC__SUCCESS` (error found) is returned, get the number of errored bits found:

```
SENSe1:ELOCation:ECOunt?
```

7 If only one errored bit is found, query the location:

```
SENSe1:BEADdress?
```

See also *"Handling the Results" on page 34* for more information.

Using Interrupts

How Serial BERT Uses Interrupts

You may want to know when a particular event occurs, without having to continually poll the reporting register. The best way to do this is with the use of interrupts.

Service Request Example

Interrupts or Service Requests (SRQ) allow the instrument to pause the controller when the contents of a particular register change. The controller can then suspend its present task, service the instrument, and return to its initial task.

The basic steps involved in generating a service request (SRQ) are as follows:

- Decide which particular event should trigger a service request.
- Locate the corresponding status register.
- Set the transition filter to pass the chosen transition of that event.

- Set the enable register from that register group to pass that event to set the summary bit in the Status Byte Register.
- Set the Status Byte Enable Register to generate an SRQ on the chosen summary bit being set.

Using Interrupts – Procedures

The process of using interrupts is best explained by looking at an actual example. The following examples generate an SRQ from an event in the Operation Status group.

Specifically, they cause the error detector to generate a service request at the end of a measurement period. This is done by catching the Serial BERT's GATE END event (bit 9 in the Operation Status register).

Using Interrupts with SCPI

To generate an interrupt at the end of a measurement period:

1 Set the Operation Status register's transition registers so that the positive transition of the GATE END bit is caught.

```
:STATus:OPERation:PTRAnsition 512
:STATus:OPERation:NTRAnsition 0
```

Note: The default setting of the transition registers is to pass only positive transitions.

2 Enable bit 9 of the Operation Status register.

```
:STATus:OPERation:ENABle 512
```

3 Program the Service Request Enable Register to generate a service request when the Operation Status summary bit (OPER) is set in the Status Byte register.

```
*SRE 128
```

Working With User Patterns

The following topics provide information on the recommended techniques for working with user patterns.

Techniques for Editing User Patterns

The recommended way to edit a user pattern in IVI-COM is as follows:

- Define the pattern

  This includes the length, description, and whether the pattern is alternate or standard.

- Insert the data

  The data format and the data itself must be defined.

- Send the pattern to the pattern generator and/or error detector
- Set up a trigger on the pattern generator to be sent with the pattern

NOTE Serial BERT can use 12 user patterns (UPATtern<n>), and any number of user pattern files (UFILe). There is absolutely no difference between these patterns. The user patterns are stored in the same format on the file system, with the name UPAT<n>.ptrn (for example, upat12.ptrn).

The user patterns are provided for backwards compatibility. It is recommended that you use the user pattern files, and *not* the user patterns.

NOTE UPAT0 is a synonym for the pattern currently executed by the instrument.

How the Serial BERT Uses Alternate Patterns

These patterns are used to define the pattern generator's data output signal. Various commands can be used to define which pattern is sent at any one time. These commands, and how they interact, are described below.

Source The source defines how the Serial BERT determines what should be output. The following alternatives are available:

- Internal

  Alternate pattern output is determined internally by the instrument (for example, from the user interface or remote program).

- External

  Alternate pattern output is determined by the signal at AUX IN. This can either be edge-sensitive or level-sensitive.

- Blanking

  Output can be shut off according to the level at AUX IN. If AUX IN high, output is generated, if AUX IN low, no output.

NOTE It is important to understand this setting regarding the usage of the signal at the AUX IN port. If you select External, the signal at AUX IN defines *which* pattern (A or B) is sent. If you select Blanking, the signal at AUX IN defines *if* a signal is sent at all. The latter also works for standard (not alternating) patterns.

Mode Mode defines how the output is generated. Alternatives are:

- Alternate

  Output signal is defined by the selected pattern.

- Oneshot

  One instance of pattern B is inserted into the output pattern upon trigger.

- LLevel

  Output depends on signal level at AUX IN. If high, pattern B is output, if low, pattern A is output.

- REdge

  Output depends on the signal edge at AUX IN. At rising edge, one instance of pattern B is output.

AlternatePattern / Select

AlternatePattern/Select defines what pattern is output. It is only applicable to Alternate patterns with the Source set to INTErnal or output Blanking. The following options are available:

- A Half

  Only pattern A is output.

- B Half

  Only pattern B is output.

- AB Half

  Pattern A and pattern B are sent alternatively (one instance A, one instance B, and so on).

The following table shows how these commands work together:

Source	Mode	Description
External	LLevel	The signal at **Aux In** controls which half of the pattern is output. If **Aux In**=logic high, pattern B is sent. If **Aux In**=logic low, pattern A is sent.
	REdge	When a rising edge occurs at the **Aux In**, a single occurrence of pattern B is inserted into a continuous pattern A output.
Internal	Alternate	The AlternatePattern / Select command controls which half of the pattern is output. The options are: pattern A only (A half) pattern B only (B half) alternating A B (AB half)
	Oneshot	IVI-COM: The BShot command inserts one instance of pattern B into the output. SCPI: The :APCHange:IBHalf command inserts one instance of pattern B into the output.
Blanking	Alternate	The signal at **Aux In** controls whether output is generated: If **Aux In**=logic high, output is generated. If **Aux In**=logic low, no output is generated. The generated output depends on the Select command (A Half, B half, AB Half).

How Serial BERT Sends Triggers

The Serial BERT can repeatedly send trigger signals either according to a clock divider, or according to the output pattern.

Triggering upon Divided Clock

The trigger pulse is sent from the pattern generator's TRIG OUT port. If the trigger mode is **Divided Clock**, the trigger is sent according to the clock ratio.

Triggering upon Pattern

If the trigger mode is **Pattern**, the trigger is sent according to the selected pattern.

Depending on the selected pattern, you have the following possibilities for setting the position of the trigger:

- PRBS and PRBN patterns

  You can define the pattern, the occurrence of which sends the trigger.

- Mark Density and Zero Substitution patterns

  You can define the bit position that causes a trigger to be sent.

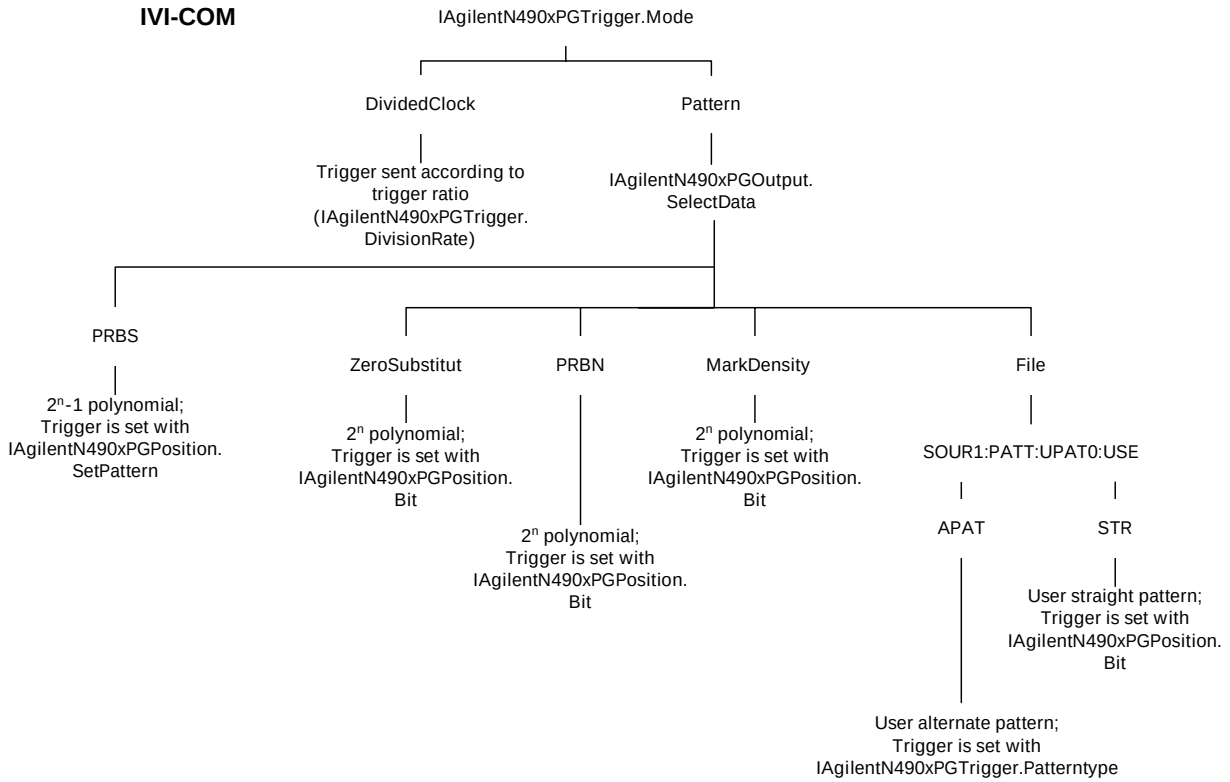
- User patterns

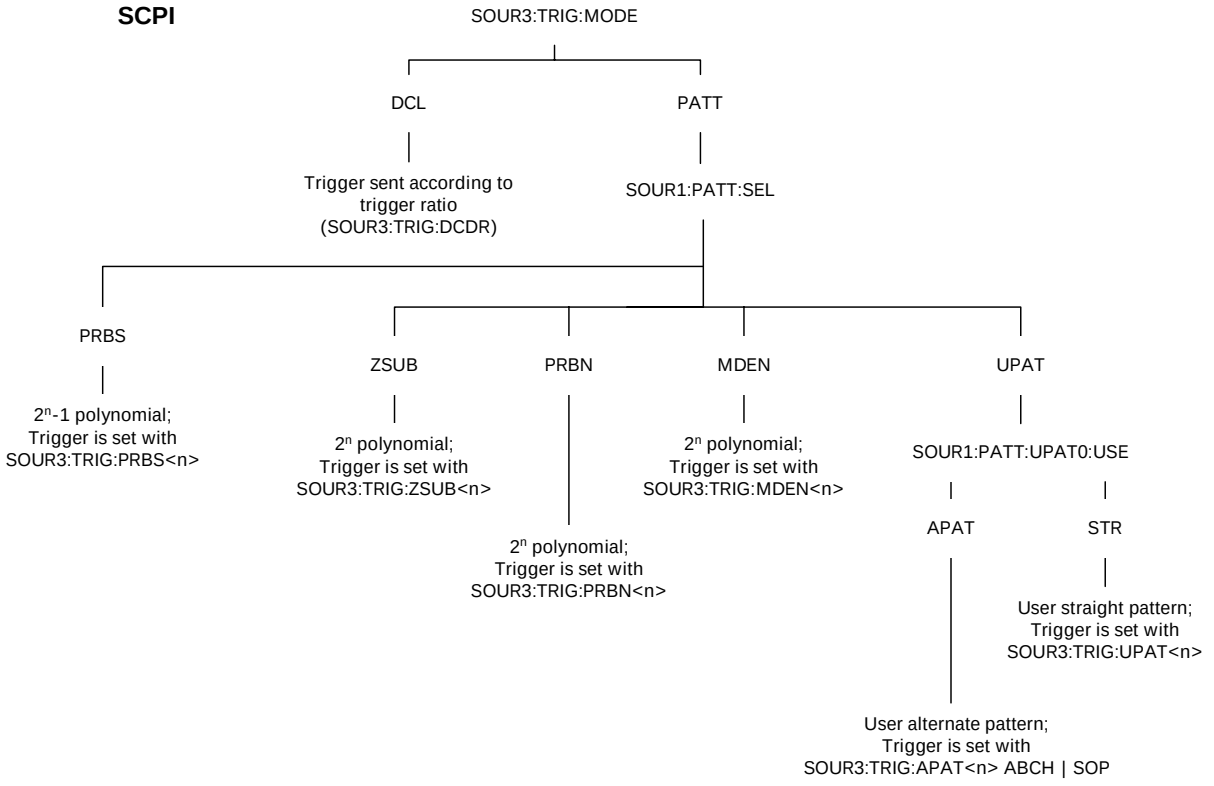
  You can define whether a trigger is sent every time a pattern is sent, or every time a pattern is changed (for alternate patterns).

Triggering upon Alternate Patterns

Alternate patterns are composed of two halves. The half that is sent out can be defined according to input at the Aux In port, triggered by the instrument internally, or can be triggered by the user. This is defined according the mode.

The following graphics shows the dependencies for sending patterns.





Working With User Patterns – Procedures

The following topics show you how to set up a program in IVI-COM and SCPI that does the following:

- Sets up an alternate pattern file and sends it to the pattern generator and error detector
- Sends triggers according to the input at AUX IN
- Sends a PRBS pattern to the pattern generator and error detector

Working with User Patterns in IVI-COM

Creating Alternate Patterns

The following code provides an example of how to set up an alternate pattern.

```
Private Sub DefinePatternFile()
' Define the classes;
' myBERT is the already created SerialBERT object
Dim myPG As IAgilentN490xPG
Dim myED As IAgilentN490xED
Dim myPGTrig As IAgilentN490xPGTrigger
Dim myPatternFile As IAgilentN490xPGPatternfile
```

```
Dim myData1() As String
Dim myData2() As String
Dim ix As Integer

Set myPG = myBERT.PGs.Item("PG1")
Set myED = myBERT.EDs.Item("ED1")
Set myPGTrig = myPG.Trigger
Set myPatternFile = myPG.Patternfiles.Item("PGPatternfile1")

' Set up one array with alternating 1s and 0s
' and one with only 0s
ReDim myData1(32)
ReDim myData2(32)
For ix = 1 To 32
   If (ix And 2) = 2 Then
      myData1(ix) = "1"
   Else
      myData1(ix) = "0"
   End If
   myData2(ix) = "0"
Next

With myPatternFile
   ' Define the pattern
   .Length = 32
   .Description = "Test pattern"
   .Alternate = True

   ' Set the pattern's data
   .SetData 1, AgilentN490xPGPatternFormatBin, myData1
   .SetData 2, AgilentN490xPGPatternFormatBin, myData2

   ' Error detector should track the pattern generator
   myED.Input.DataIns.Item("EDDataIn1"). _
         TrackingEnabled = True
   ' Now send the pattern to the instrument
   .SelectData

End With

' And finally send a trigger upon pattern change
myPGTrig.Mode = AgilentN490xPGTriggerModePattern
myPGTrig.Patterntype = AgilentN490xPGTriggerPatterntypeABChange
myPGTrig.Position.Bit = 32
End Sub
```

Triggering on AUX IN

The following example shows how to set up the Serial BERT to send pattern B upon a rising edge at AUX IN:

```
Private Sub AlternatePatterns()
' Define the classes;
' myBERT is the already created Serial BERT object
```

```
Dim myPG As IAgilentN490xPG
Dim myPGAuxIn As IAgilentN490xPGAuxIn
Dim myPGDataOut As IAgilentN490xPGOutput
Dim myPatternFile As IAgilentN490xPGPatternfile

Set myPG = myBERT.PGs.Item("PG1")
Set myPGAuxIn = myPG.Input.AuxIn
Set myPGOut = myPG.Outputs.Item("PGOutput1")

' Now send the pattern generator's
' pattern file 1 to the pattern generator
Set myPatternFile = myPG.Patternfiles.Item("PGPatternfile1")
myPatternFile.SelectData

' Set the source to be external
myPGAuxIn.Source = AgilentN490xPGAuxInSourceExternal

' We want alternate patterns
myPGAuxIn.AlternatePattern = _
      AgilentN490xPGAuxInAlternatePatternABHalf

' With B sent at the rising edge
myPGAuxIn.Mode = AgilentN490xPGAuxInModeREdge

End Sub
```

PRBS Patterns

The following code shows you how to set up a PRBS pattern and send it to the instrument:

```
Private Sub UsePRBS()
Dim myPG As IAgilentN490xPG
Dim myPGOut As IAgilentN490xPGOutput
Dim myPGTrig As IAgilentN490xPGTrigger
Dim myPGTrigPos As IAgilentN490xPGPosition
Dim myED As IAgilentN490xED
Dim myPattern() As String
Dim ix As Integer

Set myPG = myBERT.PGs("PG1")
Set myED = myBERT.EDs("ED1")
Set myPGOut = myPG.Outputs.Item("PGOutput1")
Set myPGTrig = myPG.Trigger
Set myPGTrigPos = myPG.Trigger.Position

myED.Input.DataIns.Item("EDDataIn1").TrackingEnabled = True
myPGOut.SelectData AgilentN490xPGOutputSelectPRBN, "7"

' Create an array for the trigger pattern
' We want to trigger on "0011111"
ReDim myPattern(7)
myPattern(1) = "0"
myPattern(2) = "0"
For ix = 3 To 7
   myPattern(ix) = "1"
Next
```

```
' And set the trigger
myPGTrig.Mode = AgilentN490xPGTriggerModePattern
myPGTrig.Position.SetPattern myPattern
End Sub
```

Working with User Patterns in SCPI

When creating user patterns in SCPI, it is necessary to format the data. You can use the PATTern:FORMat[:DATA] command to define the format for entering the data. This command allows you to define how the block data should be entered: as standard ASCII data (256 characters), hex data (4 bits per character), or binary data (1s and 0s).

Editing Straight Patterns

For user patterns in the STRaight mode, it is recommended that the following commands be executed *in order*:

1 Define that a STRaight pattern be used.

```
SOURce1:PATTern:UPATtern<n>:USE STRaight
```

2 Set the length of the pattern.

```
SOURce1:PATTern:UPATtern<n>[:LENGth] <NR1>
```

3 Define how the data is to be packed.

```
SOURce1:PATTern:FORMat:DATA PACKed, 1|4|8
```

4 Define the pattern data.

```
SOURce1:PATTern:UPATtern<n>:DATA <block data>
```

Defining a Trigger

Note that you can optionally define a trigger for a specific bit in the pattern:

1 Define the trigger out mode.

```
SOURce3:TRIGger:MODE PATTern
```

2 Set the bit on which the trigger is sent.

```
SOURce3:TRIGger:UPATtern<n> <NR1>
```

Editing Alternate Patterns

For user-patterns in the APATtern mode, it is recommended that the following commands be executed *in order*:

1 Define that an Alternate PATtern be used.

```
SOURce1:PATTern:UPATtern<n>:USE APATtern
```

2 Define the length of the pattern.

```
SOURce1:PATTern:UPATtern<n>[:LENGth] <NR1>
```

3 Define how the data is to be packed.

```
SOURce1:PATTern:FORMat:DATA PACKed, 1|4|8
```

4 Define the data in pattern A.

```
SOURce1:PATTern:UPATtern<n>:DATA A, <block data>
```

5 Define the data in the pattern B.

```
SOURce1:PATTern:UPATtern<n>:DATA B, <block data>
```

Defining a Trigger

Note that you can optionally define a trigger when there is a pattern change:

1 Define the trigger out mode.

```
SOURce3:TRIGger:MODE PATTern
```

2 Optionally define a trigger when there is a pattern change.

```
SOURce3:TRIGger:APATtern<n> ABCHange
```

Using Alternate Patterns

It is recommended that the following commands be executed *in order*:

1 Select the pattern to be used. This has to be an alternate pattern.

```
SOURce1:PATTern:SELect UPATtern<n>
```

2 Define the source for switching.

```
SOURce1:PATTern:APCHange:SOURce EXTernal | INTernal | BLANking
```

3 Define the mode for switching.

```
SOURce1:PATTern:APCHange:MODE
   ALTernate | ONEShot | LLEVel | REDGe
```

4 Use the following command to define which half of the pattern should be sent.

```
SOURce1:PATTern:APCHange:SELect AHALf | BHALf | ABHAlf
```

Examples for Using User Patterns in SCPI

NOTE

When the pattern is loaded to the pattern generator, it is also loaded to the error detector (TRACking ON). Keep in mind that the error detector can only track pattern A. When pattern B is sent, the error detector will still expect pattern A.

To set up a user pattern using SCPI:

1 Set the error detector to track the pattern generator (that is, to use the same pattern).

```
SENSe1:PATTern:TRACk ON
```

2 Define the file `'ALT1s0s.ptrn'` to be an alternate pattern.

```
SOURce1:PATTern:UFILe:USE 'ALT1s0s.ptrn', APATtern
```

3 Define the input data format to be binary (1s and 0s).

```
SOURce1:PATTern:FORMat:DATA PACKed, 1
```

4 Set the pattern length to 8 bits.

```
SOURce1:PATTern:UFILe:LENGth 'ALT1s0s.ptrn', 8
```

5 Define pattern A.

```
SOURce1:PATTern:UFILe:DATA A, 'ALT1s0s.ptrn', #1810101010
```

6 Define pattern B.

```
SOURce1:PATTern:UFILe:DATA B, 'ALT1s0s.ptrn', #1800000000
```

7 Load the pattern to the pattern generator.

```
SOURce1:PATTern:SELect FILENAME, 'ALT1s0s.ptrn'
```

NOTE When the pattern is loaded to the pattern generator, it is also loaded to the error detector (TRACking ON). Keep in mind that the error detector can only track pattern A. When pattern B is sent, the error detector will still expect pattern A.

Switching at Aux In

With these commands, pattern A is sent when the input at **Aux In** is low. When the input is high, pattern B is sent.

1 Load the previously defined pattern to the pattern generator.

```
SOURce1:PATTern:SELect FILename, 'ALT1s0s.ptrn'
```

2 Select the source for switching patterns to **Aux In**.

```
SOURce1:APCHange:SOURce EXTernal
```

3 Define that alternate patterns should be sent.

```
SOURce1:APCHange:MODE ALTernate
```

Generating a Trigger

The following commands expand on the previous example. They cause a a trigger to be generated on the **Trigger Out** port whenever the user pattern is changed (from pattern A to pattern B).

1 Define the trigger output mode.

```
SOURce3:TRIGger:MODE PATTern
```

2 Set up the trigger for pattern changes.

```
SOURce3:TRIGger:APATtern ABCHange
```

Switching on the Rising Edge

With these commands, pattern A is sent until a rising edge is detected at **Aux In**. When the rising edge is detected, pattern B is sent.

1 Load the pattern to the pattern generator.

```
SOURce1:PATTern:SELect FILENAME, 'ALT1s0s.ptrn'
```

2 Set the source for switching patterns to **Aux In**.

```
SOURce1:APCHange:SOURce EXTernal
```

3 Define that pattern B should be sent upon the rising edge.

```
SOURce1:APCHange:MODe REDGe
```

Programmatically Switching

These commands allow the programmer to manually set which pattern should be sent.

1 Load the pattern to the pattern generator.

```
SOURce1:PATTern:SELect FILENAME, 'ALT1s0s.ptrn'
```

2 Select the source for changing patterns to be internal.

```
SOURce1:APCHange:SOURce INTernal
```

3 Define that alternate patterns should be sent.

```
SOURce1:APCHange:MODE ALTernate
```

4 Send pattern A continuously.

```
SOURce1:PATTern:APCHange:SELect AHALf
```

5 After some event occurs, change to pattern continuous B.

```
SOURce1:PATTern:APCHange:SELect BHALf
```

6 And then set up output to automatically alternate between pattern A and pattern B.

```
SOURce1:PATTern:APCHange:SELect ABHAlf
```

Inserting Pattern B

These commands allow one instance of pattern B to be inserted into the output when the Insert B button in the user interface is pressed.

1 Load the pattern to the pattern generator.

```
SOURce1:PATTern:SELect FILENAME, 'ALT1s0s.ptrn'
```

2 Select the source for changing patterns to be internal.

```
SOURce1:PATTern:APCHange:SOURce INTernal
```

3 Select the mode to insert a single instance of pattern B.

```
SOURce1:PATTern:APCHange:MODE ONEShot
```

4 Use *Insert B* button in GUI or use remote command in order to insert pattern B in the data output.

```
SOURce1:PATTern:APCHange:IBHalf ONCe
```

SCPI Command Language

The Serial BERT is compatible with the standard language for remote control of instruments. **S**tandard **C**ommands for **P**rogrammable **I**nstruments (SCPI) is the universal programming language for instrument control.

SCPI can be subdivided into the following command sets:

- SCPI Common Commands
- SCPI Instrument Control Commands
- IEEE 488.2 Mandatory and Optional Commands

SCPI Common Commands

This is a common command set. It is compatible with IEEE 488.2 and contains general housekeeping commands. The common commands are always headed by an asterisk. A typical example is the reset command: *RST

The IEEE 488.2 command set also contains query commands. Query commands always end with a question mark.

SCPI Instrument Control Commands

The programming commands are compatible with the Standard Commands for Programmable Instruments (SCPI) standard. For more detailed information regarding the GPIB, the IEEE 488.2 standard, or the SCPI standard, refer to the following books:

- SCPI Consortium. SCPI–Standard Commands for Programmable Instruments, 1997 (http://www.scpiconsortium.org).
- International Institute of Electrical and Electronics Engineers. IEEE Standard 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation. New York, NY, 1987.
- International Institute of Electrical and Electronics Engineers. IEEE Standard 488.2-1987, IEEE Standard Codes, Formats, Protocols and Common commands For Use with ANSI/IEEE Std 488.1-1987. New York, NY, 1987.

IEEE 488.2 Mandatory and Optional Commands

In order to comply with the SCPI model as described in IEEE 488.2, the Serial BERT implements certain mandatory commands. Other commands are implemented optionally. For more detail on the IEEE 488.2 mandatory and optional commands, see *"Mandatory Commands" on page 63* and *"Optional Commands" on page 68*.

Overlapped and Sequential Commands

IEEE 488.2 defines the distinction between overlapped and sequential commands. A sequential command is one which finishes executing before the next command starts executing. An overlapped command is one which does not finish executing before the next command starts executing.

The Serial BERT has the following overlapped commands:

- SENSe[1]:GATE[:STATe] ON | 1

  (when GATE:MODE SINGle)
- SENSe[1]:EYE:TCENter|:TCENter ONCE | ON | 1
- SENSe[1]:EYE:ACENter|:ACENter ONCE | ON | 1
- SENSe[1]:EYE:ALIGn:AUTO ONCE | ON | 1
- SENSe[1]:EYE:QUICk:TCENter ONCE | ON | 1
- SENSe[1]:EYE:QUICk:ACENter ONCE | ON | 1
- SENSe[1]:EYE:QUICk:ALIGN:AUTO ONCE | ON | 1
- SENSe[1]:SYNChronizat ONCE
- SENSe[1]:ELOCation ONCE

NOTE It is not be reliable to use wait statements in the control program to facilitate the use of overlapped commands.

Because these commands may allow the execution of more than one command at a time, special programming techniques must be used to ensure valid results. The common commands *OPC, *WAI, and *OPC? can be used for this purpose. They help synchronize a device controller with the execution of overlapped commands.

The behaviors of these commands, in brief, are as follows:

- *OPC

  The *OPC command sets the Operation Complete (OPC) bit of the Standard Event Status Register (SESR) when the No Operation Pending flag is TRUE (No Operation Pending flag is attached to

each overlapped command). Until that time, the controller may continue to parse and execute previous commands. It is good technique, then, to periodically poll the OPC bit to determine if the overlapped command has completed.

- *WAI

  The *WAI command allows no further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.

- *OPC?

  The *OPC? query returns the ASCII character "1" in the Output Queue when the No Operation Pending flag is TRUE. At the same time, it also sets the Message Available (MAV) bit in the Status Byte Register. The *OPC? will not allow further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.

NOTE The command behaviors described above are for overlapped commands. When the same commands are used with sequential commands, the behaviors may be different.

Operation Pending Events

For the Serial BERT, six conditions can change an operation pending flag. Notice that the first four correspond to the four overlapped commands:

- A single timed accumulation period has expired.
- The automatic eye-time-centering operation has expired.
- The automatic eye-amplitude-centering operation has expired.
- An automatic alignment has occurred.
- The requested operation failed.
- The operation was aborted by the user.

Data Types

The Serial BERT has the capability of receiving and returning data in the following formats:

- STRING

  A string of human-readable ASCII characters, either quoted or non-quoted.

- NUMERIC

  The Serial BERT handles three numeric formats:

  - **<NR1>**: Integer (0, 1, 2, –1, etc.)
  - **<NR2>**: Number with an embedded decimal point (0.1, 0.001. 3.3, etc.)
  - **<NR3>**: Number with an embedded decimal point and exponent (1e33, 1.3e–12, etc.)
  - Hex preceded by #h (#hff, #hFF, etc.)

- BOOLEAN

  Boolean values can be sent to the Serial BERT as either TRUE | FALSe or 0 | 1. The Serial BERT answers queries with 0 | 1.

- BLOCK DATA

  Block data is used when a large quantity of related data is being returned. A definite length block is suitable for sending blocks of 8-bit binary information when the length is known beforehand. An indefinite length block is suitable for sending blocks of 8-bit binary information when the length is not known beforehand or when computing the length beforehand is undesirable.

  It has the following format:

  #<Length of length><Length of data><data>

  <Length of length> is a single integer that contains the number of digits in <Length of data>, which in turn contains the length of the data. So, for example, a 512-byte pattern would be defined as:

  #3512<data>

Important Points about SCPI

There are a number of key areas to consider when using SCPI for the first time. These are as follows:

- Instrument Model
- Command Syntax
- Optional Parts of Commands
- Sending Commands
- Command Separators
- SCPI Command Structure

Instrument Model

SCPI guidelines require that the Serial BERT is compatible with an instrument model. This ensures that when using SCPI, functional compatibility is achieved between instruments that perform the same tasks. For example, if two different instruments have a programmable clock frequency setting, then both instruments would use the same SCPI commands to set their frequency. The instrument model is made up of a number of subsystems.

The sub-system defines a group of functions within a module and has a unique identifier under SCPI, which is called the Root Keyword.

For more details on the instrument model, see *"Serial BERT Register Model" on page 25*.

Command Syntax

Commands may be up to twelve characters long. A short-form version is also available which has a preferred length of four characters or less. In this document the long-form and short-form versions are shown as a single word with the short-form being shown in upper-case letters.

For example, the long-form node command SOURce has the short-form SOUR. Using the short form saves time when entering a program, however, using the long form makes a program more descriptive and easier to understand.

SCPI commands may be commands only, commands and queries, or queries only. A question mark at the end of a command indicates that it is a query. If the question mark appears in brackets ([?]), the command has a command and query form.

Optional Command Keywords

Some layers in the SCPI command structure are optional. These optional keywords are indicated by square brackets ([ ]). A typical use for these types of keywords is with a command that is unique to one module. In this case, the top layer (Root Keyword) of the command structure may be omitted.

For example, the following command code segments are functionally identical:

```
[SOURce[1]:]PATTern:MDENsity[:DENSity] <numeric value>
SOURce:PATTERN:MDENSITY <numeric value>
PATTern:MDENsity <numeric value>
```

```
PATT:MDEN <numeric value>
patt:mden <numeric value>
```

Note that it is not necessary to include the syntax inside the square brackets ([ ]).

Sending Commands

Commands are sent over the GPIB in the same way that GPIB and IEEE 488.2 common commands are sent. The difference is that the SCPI command is "nested" into the programming language of choice. The programming language of choice may be a language such as Visual Basic, C++, or SICL.

For an examples of how commands are sent, see *"Sending Commands to the Serial BERT" on page 58*.

Querying Responses

It is possible to interrogate the individual settings and status of a device using query commands. Retrieving data is a two-stage operation.

The query command is sent from the controller using the OUTPUT statement and the data is read from the device using the ENTER statement. A typical example, using the SCPI IEEE 488.2 Common Command *IDN? which queries the identity of a device.

See *"Sending Commands using VISA" on page 58* for an example in the C programming language of how to query the identity.

NOTE When sending strings to the instrument, either the double quote (") or the single quote may be used ('), the former being more suited to PASCAL programs, which make use of a single quote; the latter being more suited to use in BASIC programs, which uses a double quote as a delimiter. In this manual, the double quote has been used throughout.

Command Separators

The SCPI command structure is hierarchical and is governed by commas, semi-colons and colons:

- Commas are used to separate parameters in one command.
- Colons are used to separate levels.
- Semi-colons are used to send more than one command to the instrument at a time.

```
SENSe[1]:PATTern:UPATtern<n>:IDATa [A|B,]
<start_bit>, <length_in_bits>,<block_data>
```

Note that the command hierarchy is indicated by colons and that the parameters (beginning with [A|B,]), are separated by commas.

Multiple Commands

It is possible to send several commands in one pass, as long as the commands all belong to the same node in the SCPI tree. The commands have to be separated by semicolons.

The following SCPI commands provide examples of this. Note that the optional characters and keywords have been removed.

```
SOURce1:VOLTage:LEVel:IMMediate:OFFSet 1.5
SOURce1:VOLTage:LEVel:IMMediate:AMPLitude 2
```

These commands can also be sent as follows:

```
VOLT:OFFS 1.5; AMPL 2.0
```

SCPI Command Structure Example

The SCPI command structure can be best examined by means of an example. For example, the command to select the pattern generator's pattern is:

```
[SOURce[1]]:PATTern[:SELect] PRBS7
```

The structure of this command can be illustrated as follows:

[SOURce [1]:]	This is the top layer of the command structure and identifies the pattern generator source subsystem.
PATTern	This is the next layer and defines subnode for setting up the pattern.
[:SELect]	This is the command itself, and is the equivalent of setting the front panel pattern selection field.
PRBS(n)	This is the parameter required by the PATTern command keyword.

NOTE Any optional commands are enclosed in square brackets [ ] and any optional characters are shown in lower case.

A colon indicates a change of level in the command hierarchy. Commands at the same level in the hierarchy may be included in the same command line, if separated by a semi-colon.

The bar symbol (|) indicates mutually exclusive commands.

To translate this syntax into a command line, follow the convention described above. Remember, however, that the command line can be created in several different ways. It can be created with or without optional keywords, and in a long or short form. The following example gives three possible forms of the command line; all are acceptable.

In long form:

```
SOURce1:PATTern:SELect PRBS7
```

In short form:

```
SOUR1:PATT:SEL PRBS7
```

With the optional commands removed:

```
PATT PRBS7
```

The long form is the most descriptive form of programming commands in SCPI. It is used for the examples in this manual.

Sending Commands to the Serial BERT

A command is invalid and will be rejected if:

- It contains a syntax error.
- It cannot be identified.
- It has too few or too many parameters.
- A parameter is out of range.
- It is out of context.

Sending Commands using VISA

The following code example shows how to use the VISA library to connect to the instrument via GPIB. This code also contains commented examples for USB and LAN.

This example queries the device for the identification string and prints the results.

```
#include <visa.h>
#include <stdio.h>
```

```
void main () {
   ViSession defaultRM, vi;
   char buf [256] = {0};

   /* Open session to GPIB device at address 14 */
   viOpenDefaultRM (&defaultRM);
   viOpen (defaultRM, "GPIB0::14::INSTR", VI_NULL,VI_NULL, &vi);

   /* Alternatively open a session to the device at
      IP address 10.0.1.255 */
   /* viOpen (defaultRM,
      "TCPIP0::10.0.1.255::INSTR", VI_NULL,VI_NULL, &vi); */

   /* Or open a session to the USB device */
   /* viOpen (defaultRM,
            "usb0[2391::20496::SNN4900AXXXDE::0::INSTR]",
            VI_NULL,VI_NULL, &vi); */

   /* Or if you have assigned an alias N4901A-Lab */
   /* viOpen (defaultRM, "N4901A-Lab", VI_NULL, VI_NULL, &vi); */

   /* Initialize device */
   viPrintf (vi, "*RST\n");

   /* Send an *IDN? string to the device */
   viPrintf (vi, "*IDN?\n");

/* Read results */
   viScanf (vi, "%t", &buf);

/* Print results */
   printf ("Instrument identification string: %s\n", buf);

/* Close session */
   viClose (vi);
   viClose (defaultRM);
}
```

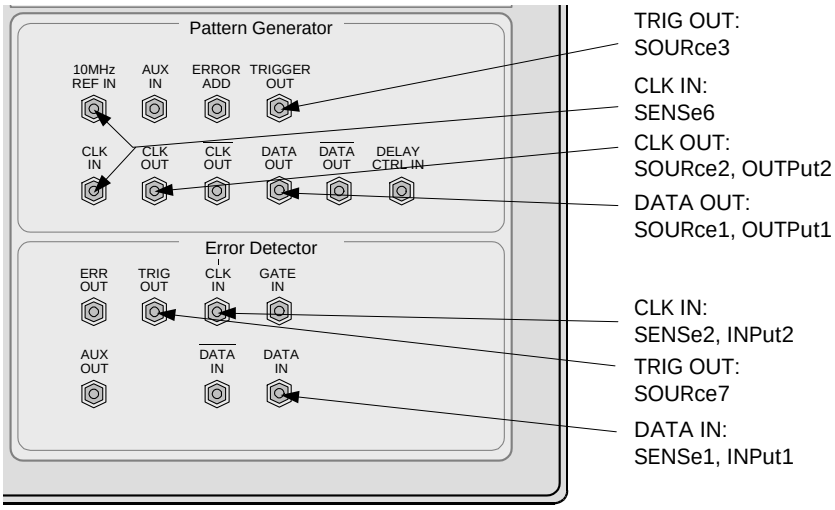
This returns the identity string
“AGILENT TECHNOLOGIES,N4901A,3331U00101,A.01.01”.

SCPI Command Reference

Serial BERT Subsystems

TIP You can use the Output Window in the instrument's user interface to monitor the SCPI commands and queries. This can make it easier to find out which command is responsible for which action.

The SCPI commands are divided into *subsystems*, which reflect various functionality of the instrument. The following figure shows where the port-related subsystems are located.



The SOURce subsystems control output signals (for example, for defining output patterns and levels). The OUTPut subsystems control the electrical port connection (for example, to disconnect the port or set the terminations).

The SENSe subsystems control the expected input signal. They correspond to the SOURce subsystems. The INPut subsystems correspond to the OUTPut subsystems; they are responsible for the electrical port connection.

NOTE The inverted clock and data outputs track the standard outputs. For example, the pattern generator's $\overline{\text{DATA OUT}}$ port tracks the DATA OUT port. Any changes to the standard output automatically modifies the inverted output (and vice versa). Therefore, only the commands of the standard outputs are documented here.

Besides the subsystems shown above, the following subsystems are available:

- STATus

  This subsystem controls the SCPI-compatible status reporting structures.

  IVI-COM Equivalent: IAgilentN490xStatus

- SYSTem

  This subsystem controls functions such as general housekeeping and global configurations.

  IVI-COM Equivalent: IAgilentN490xSystem

- TEST

  This subsystem verifies specific hardware components for basic functionality.

  IVI-COM Equivalent: IIviDriverUtility.SelfTest

All subsystems commands are described in this chapter.

IEEE Commands

Mandatory Commands

The following *mandatory* IEEE 488.2 commands are implemented:

Name	Description under
*CLS	*"*CLS" on page 63*
*ESE[?]	*"*ESE[?]" on page 64*
*ESR?	*"*ESR?" on page 64*
*IDN?	*"*IDN?" on page 64*
*OPC	*"*OPC" on page 65*
*OPC?	*"*OPC?" on page 65*
*RST	*"*RST" on page 66*
*SRE[?]	*"*SRE[?]" on page 67*
*STB?	*"*STB?" on page 67*
*TST?	*"*TST?" on page 67*
*WAI	*"*WAI" on page 68*

*CLS

IVI-COM Equivalent IAgilentN490xStatus.Clear (not IVI-compliant)

Syntax *CLS

Description This command clears all status data structures in a device. For the Serial BERT, these registers include:

SESR	IEEE 488.2
OPERation Status Register	SCPI
QUEStionable Status Register	SCPI

Execution of *CLS also clears any additional status data structures implemented in the device. The corresponding enable registers are unaffected.

See *"Serial BERT Register Model" on page 25* for more information about the Status Byte.

*ESE[?]

Syntax *ESE <Num.>

*ESE?

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Description

The Standard Event Status Enable Command (*ESE) sets the Standard Event Enable Register. This register acts like a mask, so that the next time a selected bit goes high, the ESB bit in the status byte is set. See *"Serial BERT Register Model" on page 25* for details.

For example, if bit 0 is set in the Standard Event Enable Register, then when the OPC bit in the Standard Event register goes true, the ESB summary bit is set in the Status Byte.

The query (*ESE?) returns the contents of the Standard Event Enable Register.

*ESR?

IVI-COM Equivalent IAgilentN490xStatus.SerialPoll (not IVI-compliant)

Syntax *ESR?

Description This query interrogates the Standard Event Status Register. The register is cleared after it is read.

*IDN?

IVI-COM Equivalent IIviDriverIdentity (IVI-compliant)

Syntax *IDN?

Description For the Serial BERT, the Identification Query (*IDN?) response semantics are organized into four fields, separated by commas. The field definitions are as follows:

Field	Value
Manufacture	Agilent Technologies
Model	N4906A
Serial Number	DExxxxxxxx
Firmware Level	A.x.x.xxx

*OPC

Syntax *OPC

Description A device is in the Operation Complete Command Active State (OCAS) after *OPC has been executed. The device returns to the Operation Complete Command Idle State (OCIS) whenever the No Operation Pending flag is TRUE, while at the same time setting the OPC bit of the ESR TRUE.

The following events force the device into OCIS without setting the No Operation Pending flag to TRUE and without setting the OPC bit of the ESR:

- power on
- receipt of a DCAS message (device clear)
- execution of *CLS
- execution of *RST

Implementation of the *OPC command is straightforward in devices that implement only sequential commands. When executing *OPC, the device simply sets the OPC bit of the ESR.

In devices that implement overlapped commands, the implementation of *OPC is more complicated. After executing *OPC, the device must not set the OPC bit of ESR until the device returns to OCIS, even though it continues to parse and execute commands.

NOTE For the Serial BERT, *OPC can be used with overlapped commands. For more information, see *"Overlapped and Sequential Commands" on page 52*.

*OPC?

IVI-COM Equivalent IAgilentN490xSystem.WaitForOperationComplete (not IVI-compliant)

Syntax *OPC? Command

Description A device is in the Operation Complete Query Active State (OQAS) after it has executed *OPC?. The device returns to the Operation Complete Query Idle State (OQIS) whenever the No Operation Pending flag is TRUE, at the same time placing a "1" in the Output Queue.

The following events force the device into OQIS without setting the No Operation Pending flag TRUE and without placing a “1” in the Output Queue:

- power on
- receipt of the dcas message (device clear)

Implementation of the *OPC? query is straightforward in devices which implement only sequential commands. When executing *OPC? the device simply places a “1” in the Output Queue.

The implementation of overlapped commands in a device complicates the implementation of *OPC? and places some restrictions on the implementation of the Message Exchange Protocol (MEP). IEEE 488.2 dictates that devices shall send query responses in the order that they receive the corresponding queries. Although IEEE 488.2 recommends that *OPC? be the last query in a program message, there is nothing to prevent a controller program from ignoring this suggestion. This is why *OPC? must be sequential.

NOTE For the Serial BERT, *OPC(?) can be used with overlapped commands. For more information, see *“Overlapped and Sequential Commands” on page 52*.

*RST

IVI-COM Equivalent IIviDriverUtility.Reset (IVI-compliant)

Syntax *RST

Description The Reset Command (*RST) sets the device-specific functions to a known state that is independent of the past-use history of the device. The command has the same effect as the front-panel PRESET key.

In addition, receipt of *RST by the error detector will cause all past results to be reset to zero.

*SRE[?]

IVI-COM Equivalent IAgilentN490xStatus.ConfigureServiceRequest (not IVI-compliant)

Syntax *SRE <Num.>

*SRE?

Description The Service Request Enable Command (*SRE) sets the Service Request Enable Register. This acts as a mask on the Status Byte, defining when the instrument can issue a service request. For a service request to be issued, the summary bit in the Status Byte must match the bit in the Service Request Enable Register. More than one bit may be set by the *SRE command.

The query returns the current contents of the Service Request Enable Register.

See *"Serial BERT Register Model" on page 25* for details.

*STB?

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax *STB?

Description The Read Status Byte Query (*STB?) allows the programmer to read the status byte and Master Summary Status bit. When the status byte is read using the *STB command, bit 6 of the status byte is referred to as the Master Summary (MSS) bit. With this query, the status byte is not cleared when the value is read. It always reflects the current status of all the instrument's status registers.

See *"Serial BERT Register Model" on page 25* for details.

*TST?

IVI-COM Equivalent IIviDriverUtility.SelfTest (not IVI-compliant)

Syntax *TST?

Description The self-test query starts all internal self-tests and places a response into the output queue indicating whether or not the device completed the self-tests without any detected errors. It returns a 0 for success; a 1 if a failure was detected.

Upon successful completion of *TST?, the device settings are restored to their values prior to the *TST?

For more precise self-test results, use *"TEST:EXECute?" on page 181*.

*WAI

Syntax *WAI

Description The *WAI commands allows no further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.

The *WAI command can be used for overlapped commands. It stops the program execution until any pending overlapped commands have finished. Specifically, it waits until the No Operation Pending flag is TRUE, or receipt of a dcas message, or a power on.

Optional Commands

The following *optional* IEEE 488.2 commands are implemented:

Command	Description
*OPT?	Option Identification Query
*PSC	Power On Status Clear Command
*PSC?	Power On Status Clear Query
*RCL	Recall device setup
*SAV	Save device setup

*OPT?

Syntax *OPT?

Description The Option Identification query is for identifying reportable device options over the system interface.

This is a standard SCPI command. Please refer to the SCPI specification for details.

*PSC

Syntax *PSC

Description The Power-on Status Clear command controls the automatic power-on clearing of the Service Request Enable Register, the Standard Event Status Enable Register, and the Parallel Poll Enable Register.

This is a standard SCPI command. Please refer to the SCPI specification for details.

*RCL

IVI-COM Equivalent IAgilentN490xSystem.RecallState (IVI-compliant)

Syntax *RCL <numeric value | string>

Description This command loads the setup from a numbered store or from a full path filename that was previously stored with *"*SAV" on page 69*. The range of store numbers is 0 through 9.

In addition, upon receipt of *RCL, the error detector will reset all past results to zero.

NOTE Depending on the patterns that are saved with the setup, the instrument may require up to half a minute to settle. See *"Determining if Conditions have Settled" on page 21* for details.

*SAV

IVI-COM Equivalent IAgilentN490xSystem.SaveState (IVI-compliant)

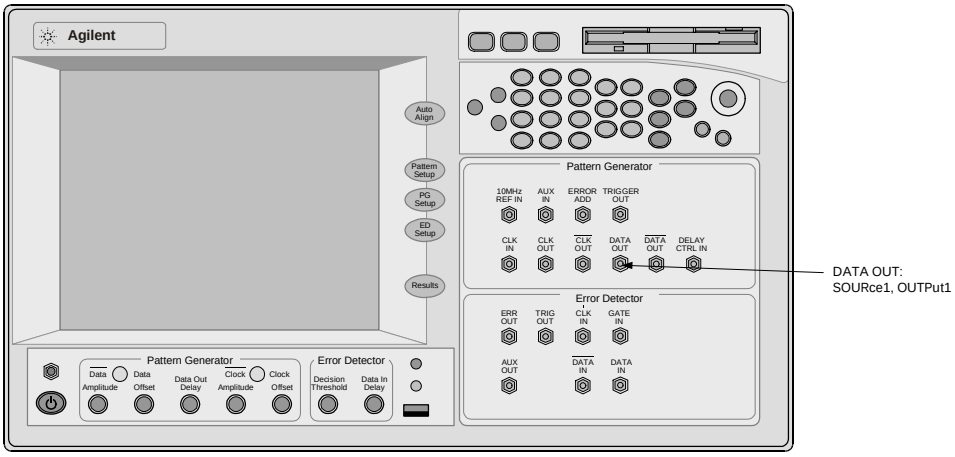
Syntax *SAV <numeric value | string>

Description This command saves the current instrument setup into a numbered store or into a full path filename. The range of store numbers is 0 through 9. The *"*RCL" on page 69* restores the setup.

The setup saves the currently used patterns, signal definitions, and other user interface settings.

SOURce[1] Subsystem

The SOURce[1] subsystem controls the pattern generator's Data Out port.



This subsystem has the following SCPI structure:

```
[:SOURce[1]:]
  :PM
    [STATe][?]
  PATTern
    . . .
  :VOLTage
    . . .
```

This subsystem has the following commands and subnodes:

Name	Description under
Commands	
:PM	*"[SOURce[1]]:PM[:STATe][?]" on page 71*
Subnodes	
:PATTern	*"[SOURce[1]]:PATTern Subnode" on page 71*
:VOLTage	*"[SOURce[1]]:VOLTage Subnode" on page 89*

[SOURce[1]]:PM[:STATe][?]

IVI-COM Equivalent IAgilentN490xPGDelayControlInput.Enabled (not IVI-compliant)

Syntax [SOURce[1]]:PM:STATe ON | OFF | 0 | 1

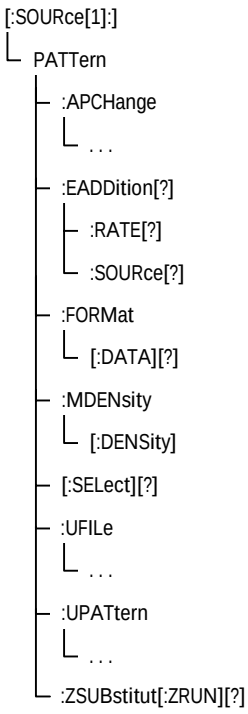
[SOURce[1]]:PM:STATe?

IVI-COM Equivalent Agt8613xBERT.PatternGenerator.DataOutput

Description Enables/disables delay control input. The query returns the state of the delay control input (0 | 1).

[SOURce[1]]:PATTern Subnode

This subnode has the following SCPI structure:



This subnode has the following commands and subnodes:

Name	Description under
Commands	
:EADDition[?]	*"[SOURce[1]]:PATTern:EADDition[?]" on page 72*
:EADDition:RATE[?]	*"[SOURce[1]]:PATTern:EADDition:RATE[?]" on page 73*

Name	Description under
:EADDition:SOURce[?]	*"[SOURce[1]]:PATTern:EADDition:SOURce[?]" on page 73*
:FORMat[:DATA][?]	*"[SOURce[1]]:PATTern:FORMat[:DATA][?]" on page 73*
:MDENsity[:DENSity][?]	*"[SOURce[1]]:PATTern:MDENsity[:DENSity][?]" on page 74*
[:SELect][?]	*"[SOURce[1]]:PATTern[:SELect][?]" on page 74*
:ZSUBstitut[:ZRUN][?]	*"[SOURce[1]]:PATTern:ZSUBstitut[:ZRUN][?]" on page 76*
Subnodes	
:APCHange	*"[SOURce[1]]:PATTern:APCHange Subnode" on page 77*
:UFILe	*"[SOURce[1]]:PATTern:UFILe Subnode" on page 80*
:UPATtern<n>	*"[SOURce[1]]:PATTern:UPATTern Subnode" on page 85*

[SOURce[1]]:PATTern:EADDition[?]

Syntax [SOURce[1]]:PATTern:EADDition <EADD>

[SOURce[1]]:PATTern:EADDition?

Input Parameters **<EADD>** ONCE | 0 | 1 | OFF | ON

Return Range 0 | 1

Description This command is a contraction of the phrase **E**rror **ADD**ition. It is used to control the addition of errors into the generated pattern.

The parameter ONCe causes a single bit error to be added to the pattern. It depends on the previous status of this command and the selected source (see *"[SOURce[1]]:PATTern:EADDition:SOURce[?]" on page 73*). The following table lists the dependencies:

:EADD	:EADD:SOUR	:EADD ONCe
0	EXT	Active
	FIX	Active
1	EXT	Active
	FIX	Not active (command has no effect)

The query returns the current state of error addition.

[SOURce[1]]:PATTern:EADDition:RATE[?]

Syntax [SOURce[1]]:PATTern:EADDition:RATE <RATE> 10^(–3, –4,... –9)

[SOURce[1]]:PATTern:EADDition:RATE?

Return Range 10^(–3, –4,... –9)

Description The command controls the rate of internal fixed error addition. Values between 10^3 and 10^9 in decade steps are permitted.

The query returns the current error add rate.

[SOURce[1]]:PATTern:EADDition:SOURce[?]

Syntax [SOURce[1]]:PATTern:EADDition:SOURce EXTernal | FIXed

[SOURce[1]]:PATTern:EADDition:SOURce?

Return Range EXT | FIX

Description The command controls the source of injected errors:

- EXTernal (and :EADDition[:STATe] is ON)

  Each pulse at the **Error Add** port causes an error to be added to the data stream.

- FIXed (and :EADDition[:STATe] is ON)

  Repetitive errors are internally added to the data stream. The rate of error addition is controlled by the :EADDition:RATE command.

The query returns the current error addition mode.

[SOURce[1]]:PATTern:FORMat[:DATA][?]

IVI-COM Equivalent Included in IAgilentN490xPGPatternfile.SetData (IVI-compliant)

Syntax [SOURce[1]]:PATTern:FORMat:DATA <PACKed>, <Num.>

[SOURce[1]]:PATTern:FORMat:DATA?

Input Parameters **<PACKed>** permits the packing of bits within a byte to be set.

<NR1> Can be 1, 4, or 8.

Return Range 1 | 4 | 8

Description The command controls the format of data transfer for the :PATTern:UPATtern<n>:DATA, :PATTern:UPATtern<n>:IDATa, :PATTern:UFILe:DATA and :PATTern:UFILe:IDATa commands. The following values are possible:

- 1

  The data is sent as a string of 1s and 0s.

- 4

  The data is sent as a string of hex characters.

- 8

  The data is sent as a string of full ASCII characters.

The query returns the current value of the data pack.

See *"Working with User Patterns in SCPI" on page 47* for descriptions on how to use the data packing.

[SOURce[1]]:PATTern:MDENsity[:DENSity][?]

IVI-COM Equivalent IAgilentN490xPGOutput.MarkDensity (not IVI-compliant)

Syntax [SOURce[1]]:PATTern:MDENsity[:DENSity] <Num.>

[SOURce[1]]:PATTern:MDENsity[:DENSity]?

Input Parameters **<NR2>** 0.125, 0.25, 0.5, 0.75, 0.875

Description The command sets the ratio of high bits to the total number of bits in the pattern. The ratio may be varied in eighths, from one to seven (eighths), but excluding three and five.

The query returns the mark density in eighths.

[SOURce[1]]:PATTern[:SELect][?]

IVI-COM Equivalent IAgilentN490xPGOutput.SelectData (IVI-compliant)

Syntax [SOURce[1]]:PATTern[:SELect] <Source>

[SOURce[1]]:PATTern[:SELect]?

Input Parameters **<Source>** PRBS<n> | PRBN<n> | ZSUBstitut<n> | MDENsity<n> | UPATtern<n> | FILename, <string>

Return Range PRBS<n> | PRBN<n> | ZSUB<n> | MDEN<n> | UPAT

Description This command defines the type of pattern being generated. The parameter is retained for backwards compatibility and may be one of the following:

PRBS<n>	<n> = 7, 10, 11, 15, 23, 31
PRBN<n>	<n> = 7, 10,11,13, 15, 23
ZSUBstitut<n>	<n> = 7, 10,11,13, 15, 23
UPATtern<n>	<n> = 1 through 12
MDENsity<n>	<n> = 7, 10,11,13, 15, 23
FILename,	<string>

ZSUBstitut **Z**ero **SUB**stitution; used for defining PRBN patterns in which a block of bits is replaced by a block of zeros. The length of the block is defined by *"[SOURce[1]]:PATTern:ZSUBstitut[:ZRUN][?]" on page 76*.

MDENsity **M**ark **DEN**sity; used for defining a PRBN pattern in which the user can set the mark density. The mark density is set with *"[SOURce[1]]:PATTern:MDENsity[:DENSity][?]" on page 74*.

UPATtern<n> **U**ser **PAT**tern; used to define the contents of a pattern store. For the Serial BERT, <n> can be 1 – 12.

FILename A parameter that allows the remote user to load a user pattern from the instrument's disk drive. This is the preferred mechanism for loading user patterns in the Serial BERT.

NOTE If the pattern generator and error detector are coupled, setting the pattern by using the SOURce1:PATTern:SELect command will cause the pattern to be set in both the pattern generator and the error detector. If the pattern generator and error detector are not coupled, then the error detector pattern must be selected using the SENSe[1]:PATTern:SELect command.

The query form returns the pattern's types in short form.

NOTE If a user-defined pattern is selected and the [:SELECT]? command is used, the response is UPAT. The particular value of <n> or the name of the file specified in the command form is not returned.

To get the path of a user pattern file, use the UFILe:NAME? command.

[SOURce[1]]:PATTern:ZSUBstitut[:ZRUN][?]

IVI-COM Equivalent IAgilentN490xPGOutput.ZeroSub (not IVI-compliant)

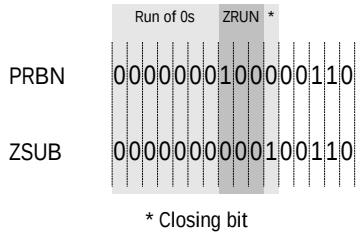
Syntax [SOURce[1]]:PATTern:ZSUBstitut[:ZRUN] MINimum | MAXimum | <numeric value>

[SOURce[1]]:PATTern:ZSUBstitut[:ZRUN]?

Return Range <NR3>

Description ZSUB patterns are PRBN patterns, where a number of bits are replaced by zeroes. The zero substitution starts after the longest runs of zeroes in the pattern (for example, for PRBN 2^7, after the run of 7 zeroes). This command allows you to define the length of the run of zeroes. For example, to produce 10 zeroes in a PRBN 2^7 pattern, three additional bits after the run of 7 zeroes must be replaced by zeroes. The bit after the run of zeroes (the closing bit) is set to 1.

The following figure shows an example, where a run of 10 zeroes is inserted into a PRBN 2^7 pattern.

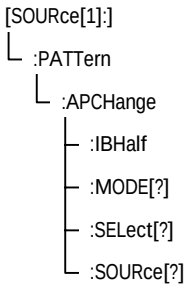


This command is only active when a ZSUB pattern has been selected (see *"[SOURce[1]]:PATTern[:SELect][?]" on page 74*).

Range The minimum value is the PRBN value. The maximum value is length of the pattern – 1. So, for a PRBN 2^7 pattern, the minimum value is 7, and the maximum value is 127 (2^7 – 1).

[SOURce[1]]:PATTern:APCHange Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:IBHalf	*"[SOURce[1]]:PATTern:APCHange:IBHalf" on page 77*
:MODE[?]	*"[SOURce[1]]:PATTern:APCHange:MODE[?]" on page 78*
:SELect[?]	*"[SOURce[1]]:PATTern:APCHange:SELect[?]" on page 79*
:SOURce[?]	*"[SOURce[1]]:PATTern:APCHange:SOURce[?] " on page 79*

[SOURce[1]]:PATTern:APCHange:IBHalf

IVI-COM Equivalent IAgilentN490xPGAuxIn.BShot (not IVI-compliant)

Syntax [SOURce[1]]:PATTern:APCHange:IBHalf ONCe

Description This command is short for **I**nsert **B Half**. It causes the insertion of a number of instances of pattern B. It is valid only when :APCHange:SOURce is set to INTernal and :APCHange:MODE is set to ONEShot. It is an event command, and as such has no query form.

Pattern B is repeated as necessary to reach the next 512-bit boundary in the memory. So, for example, if pattern B is 4 bits long, it is repeated 128 times. Or if it is 7 bits long, it is repeated 512 times.

See *"How the Serial BERT Uses Alternate Patterns" on page 40* for more information.

[SOURce[1]]:PATTern:APCHange:MODE[?]

IVI-COM Equivalent IAgilentN490xPGAuxIn.Mode (not IVI-compliant)

Syntax [SOURce[1]]:PATTern:APCHange:MODE <MODE>

[SOURce[1]]:PATTern:APCHange:MODE?

Input Parameters **<MODE>.** ALTernate | ONEShot | LLEVel | REDGe

Return Range ALT | ONES | LLEV | REDG

***RST Setting** ALTernate

Description This command controls the mode of operation of the alternate pattern output. The query returns the current mode of operation.

The parameters have the following meanings:

- ALTernate

  Alternate patterns are used. The pattern that is output must be defined with *"[SOURce[1]]:PATTern:APCHange:SELect[?]" on page 79*.

- ONEShot

  A single instance of pattern B is inserted into the output stream. This can be triggered either programmatically (with *"[SOURce[1]]:PATTern:APCHange:IBHalf" on page 77*, or from the user interface (with the *Insert B* button).

- LLEVel

  The output pattern is determined by the level of the signal at the **Aux In** port.

- REDGe

  The output pattern is determined by the rising edge of the signal at the **Aux In** port.

NOTE This command must be used together with the *"[SOURce[1]]:PATTern:APCHange:SELect[?]" on page 79* and *"[SOURce[1]]:PATTern:APCHange:SOURce[?]" on page 79*.

For instructions on how to use these commands, refer to *"How the Serial BERT Uses Alternate Patterns" on page 40*.

[SOURce[1]]:PATTern:APCHange:SELect[?]

IVI-COM Equivalent IAgilentN490xPGAuxIn.AlternatePattern (not IVI-compliant)

Syntax [SOURce[1]]:PATTern:APCHange:SELect AHALf | BHALf | ABHalf

[SOURce[1]]:PATTern:APCHange:SELect?

Return Range AHAL | BHAL | ABH

***RST Setting** AHALf

Description This command defines what pattern is output. It is only applicable to ALTernate patterns. The following options are available:

- AHALf

  Only pattern A is output.

- BHALf

  Only pattern B is output.

- ABHalf

  Pattern A and pattern B are sent alternatively (one instance A, one instance B, and so on).

This command must be used together with the *"[SOURce[1]]:PATTern:APCHange:MODE[?]" on page 78* and *"[SOURce[1]]:PATTern:APCHange:SOURce[?]" on page 79*.

For instructions on how to use these commands, refer to *"How the Serial BERT Uses Alternate Patterns" on page 40*.

The selection ABHalf is new for the Serial BERT.

[SOURce[1]]:PATTern:APCHange:SOURce[?]

IVI-COM Equivalent IAgilentN490xPGAuxIn.Source (not IVI-compliant)

Syntax [SOURce[1]]:PATTern:APCHange:SOURce EXTernal | INTernal | BLANking

[SOURce[1]]:PATTern:APCHange:SOURce?

Return Range EXT | INT | BLAN

***RST Value** EXTernal

Description This command defines how the Serial BERT determines the pattern to be output. The following alternatives are available:

- INTernal

  Alternate pattern output is determined internally by the instrument (for example, from the user interface or SCPI commands).

- EXTernal

  Alternate pattern output is determined by the signal at **Aux In**. This can either be edge-sensitive or level-sensitive.

- BLANking

  Output can be shut off according to the level at **Aux In**. If **Aux In** high, output is generated, if **Aux In** low, no output.

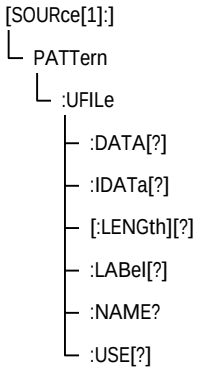
The query returns the current control of the alternate pattern output.

NOTE This command must be used together with the *"[SOURce[1]]:PATTern:APCHange:MODE[?]" on page 78* and *"[SOURce[1]]:PATTern:APCHange:SELect[?]" on page 79*.

For instructions on how to use these commands, refer to *"How the Serial BERT Uses Alternate Patterns" on page 40*.

[SOURce[1]]:PATTern:UFILe Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:DATA[?]	*"[SOURce[1]]:PATTern:UFILe:DATA[?]" on page 81*
:IDATa[?]	*"[SOURce[1]]:PATTern:UFILe:IDATa" on page 82*
[:LENGth][?]	*"[SOURce[1]]:PATTern:UFILe[:LENGth][?]" on page 83*
:LABel[?]	*"[SOURce[1]]:PATTern:UFILe:LABel[?]" on page 84*
:NAME?	*"[SOURce[1]]:PATTern:UFILe:NAME?" on page 84*
:USE[?]	*"[SOURce[1]]:PATTern:UFILe:USE[?]" on page 84*

[SOURce[1]]:PATTern:UFILe:DATA[?]

IVI-COM Equivalent IAgilentN490xLocalPatternfile.SetData (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UFILe:DATA [A|B,] <filename>, <block data>

[SOURce[1]]:PATTern:UFILe:DATA? [A|B,] <filename>

Return Range The query returns the standard (A) or alternate pattern (B) of the file found under <filename>.

Description This command is used to set the bits in user pattern files. See *"Working with User Patterns in SCPI" on page 47* for a detailed description on how to edit user patterns.

The parameters have the following meanings:

Parameter	Description
[A|B]	Defines for which pattern the data is to be set (A = standard pattern, B = alternate pattern). If the pattern file describes a standard pattern (:USE = STRaight), this parameter cannot be B.
<filename>	Name of the file being defined. If the file does not exist, it is created.
<block data>	The data that describes the pattern (see the following for the description).

<block data> The <block data> parameter contains the actual data for setting the bits of the user pattern. The bits can also be packed using the FORMat[:DATA] command. If the bits are not packed, they are handled as 8-bit data. See *"[SOURce[1]]:PATTern:FORMat[:DATA][?]" on page 73*.

This command also sets the pattern length to fit the length of the data: If the data block is longer than the pattern, the pattern is extended to fit the data; if the data block is shorter than the pattern, the pattern is truncated to the end of the data.

<block data> starts with a header that indicates the length of the desired resulting data. The length of the <block data> embedded in the header always refers to the length of the data block in bytes.

For example, consider the following header:

#19<data>

#	Start of the header.
1	Number of decimal digits to follow to form the length.
9	Length of the data block (in bytes) that follows.
<data>	The pattern data, packed according the the DATA:PACKed command.

For non-packed data (or 8-bit packed data), the <block data> required to set an 8-bit pattern of alternating 1s and 0s (01010101) would be:

#11U (Note that "U" is the ASCII representation of 85)

For 4-bit packed data, the <block data> required to set the same pattern would be:

#1255

For 1-bit packed data, the <block data> would be as follows:

#1801010101

[SOURce[1]]:PATTern:UFILe:IDATa

IVI-COM Equivalent IAgilentN490xLocalPatternfile.SetDataBlock (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UFILe:IDATa [A | B,] <filename>, <start_bit>, <length_in_bits>, <block_data>

[SOURce[1]]:PATTern:UFILe:IDATa? [A | B,] <filename>, <start_bit>, <length_in_bits>

Return Range The query returns the selected bits of the standard (A) or alternate (B) pattern of the file found under <filename>.

Description This command is used to set specific bits in a user pattern. It is similar to the :DATA command. The :IDATa command is a contraction of the phrase **I**ncremental **DAT**a and is used to download part of a user-defined pattern.

The parameters have the following meanings:

Parameter	Description
[A|B]	Defines for which pattern the data is to be set (A = standard pattern, B = alternate pattern). If the pattern file describes a standard pattern (:USE = STRaight), this parameter cannot be B.
<filename>	Name of the file being defined. If the file does not exist, it is created.
<start bit>	First bit to be overwritten (starting with 0).
<length_in_bits>	Number of bits to be overwritten.
<block data>	The data that describes the pattern (see *"[SOURce[1]]:PATTern:UFILe:DATA[?]" on page 81* for the description).

The use of the parameters can be best illustrated by an example. If we have an alternate 16-bit pattern of 0s only, and we want to set the last four bits to 1s, the IDATa command would appear as follows:

- If the data packing is 8:

  SOURce1:PATTern:UFILe:IDATa B, <filename>, 12, 4, #11(&F0)
  (where (&F0) is replaced by the ASCII representation of the value)

- If the data packing is 4:

  SOURce1:PATTern:UFILe:IDATa B, <filename>, 12, 4, #11F

- If the data packing is 1:

  SOURce1:PATTern:UFILe:IDATa B, <filename>, 12, 4, #141111

The response form returns <block data> at the specified location.

NOTE See *"Working with User Patterns in SCPI" on page 47* for more information on using this command.

[SOURce[1]]:PATTern:UFILe[:LENGth][?]

IVI-COM Equivalent IAgilentN490xLocalPatternfile.Length (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UFILe[:LENGth] <filename>, <numeric_value>

[SOURce[1]]:PATTern:UFILe[:LENGth]? <filename>

Description This command sets the length of a user pattern file. The query returns the length of the user pattern file. If an alternate pattern is selected (:USE APATtern), the LENGth command sets the length of each half of the pattern.

Note that the :DATA command automatically sets the length of the file.

See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

[SOURce[1]]:PATTern:UFILe:LABel[?]

IVI-COM Equivalent IAgilentN490xLocalPatternfile.Description (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UFILe:LABel <filename>, <string>

[SOURce[1]]:PATTern:UFILe:LABel? <filename>

Description This command sets a description for a user pattern file. The query returns the description. See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

[SOURce[1]]:PATTern:UFILe:NAME?

IVI-COM Equivalent IAgilentN490xLocalPatternfile.Location (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UFILe:NAME?

Description This query returns the file name of the currently used user pattern. It is only valid if SOURce1:PATTern:SELect? returns UPAT.

[SOURce[1]]:PATTern:UFILe:USE[?]

IVI-COM Equivalent IAgilentN490xLocalPatternfile.Alternate (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UFILe:USE <filename>, STRaight | APATtern

[SOURce[1]]:PATTern:UFILe:USE? <filename>

Return Range STR | APAT

Description This command defines whether a user pattern file should be a straight pattern or an alternate pattern:

- STRaight

  The pattern is repeatedly output.

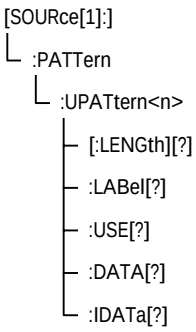
- APATtern

  The pattern is composed of two halves. The output depends on various other commands; see *"How the Serial BERT Uses Alternate Patterns" on page 40* for more information.

The default is set to have a length of 128 bits for each half pattern; all bits are set to zero and the trigger is set to occur on the A/B changeover. See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

[SOURce[1]]:PATTern:UPATTern Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
[:LENGth][?]	*"[SOURce[1]]:PATTern:UPATtern<n>[:LENGth][?]" on page 85*
:LABel[?]	*"[SOURce[1]]:PATTern:UPATtern<n>:LABel[?]" on page 86*
:USE[?]	*"[SOURce[1]]:PATTern:UPATtern<n>:USE[?]" on page 86*
:DATA[?]	*"[SOURce[1]]:PATTern:UPATtern<n>:DATA[?]" on page 87*
:IDATa[?]	*"[SOURce[1]]:PATTern:UPATtern<n>:IDATa[?]" on page 88*

NOTE For the UPATtern<n> commands, <n> can be in the range 0 – 12. 0 (zero) is used to select the current pattern, 1 – 12 selects one of the user patterns in the memory.

[SOURce[1]]:PATTern:UPATtern<n>[:LENGth][?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.Length (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UPATtern<n>[:LENGth] <numeric value>

[SOURce[1]]:PATTern:UPATtern<n>[:LENGth]?

Description This command sets the length of the selected user pattern. The query returns the length of the user pattern. If an alternate pattern is selected (:USE APATtern), the LENGth command sets the length of each half of the pattern.

Note that the :DATA command automatically sets the length of the pattern.

See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

[SOURce[1]]:PATTern:UPATtern<n>:LABel[?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.Description (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UPATtern<n>:LABel <string>

[SOURce[1]]:PATTern:UPATtern<n>:LABel?

Description The command sets the description of the pattern. The query returns the description of the pattern.

See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

[SOURce[1]]:PATTern:UPATtern<n>:USE[?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.Alternate (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UPATtern<n>:USE STRaight | APATtern

[SOURce[1]]:PATTern:UPATtern<n>:USE?

Return Range STR | APAT

Description This command defines whether a user pattern file should be a straight pattern or an alternate pattern:

- STRaight

  The pattern is repeatedly output.

- APATtern

  The pattern is composed of two halves. The output depends on various other commands; see *"How the Serial BERT Uses Alternate Patterns" on page 40* for more information.

The default is set to have a length of 128 bits for each half pattern; all bits are set to zero and the trigger is set to occur on the A/B changeover. See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

[SOURce[1]]:PATTern:UPATtern<n>:DATA[?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.SetData (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UPATtern<n>:DATA [A | B,] <block_data>

[SOURce[1]]:PATTern:UPATtern<n>:DATA? [A|B,]

Return Range The query returns the block data for pattern A or pattern B.

Description This command is used to set the bits in user pattern files. See *"Working with User Patterns in SCPI" on page 47* for a detailed description on how to edit user patterns.

The parameters have the following meanings:

Parameter	Description
[A|B]	Defines for which pattern the data is to be set (A = standard pattern, B = alternate pattern). If the pattern file describes a standard pattern (:USE = STRaight), this parameter cannot be B.
<filename>	Name of the file being defined. If the file does not exist, it is created.
<block data>	The data that describes the pattern (see the following for the description).

<block data> The <block data> parameter contains the actual data for setting the bits of the user pattern. The bits can also be packed using the FORMat[:DATA] command. If the bits are not packed, they are handled as 8-bit data. See *"[SOURce[1]]:PATTern:FORMat[:DATA][?]" on page 73*.

This command also sets the pattern length to fit the length of the data: If the data block is longer than the pattern, the pattern is extended to fit the data; if the data block is shorter than the pattern, the pattern is truncated to the end of the data.

<block data> starts with a header that indicates the length of the desired resulting data. The length of the <block data> embedded in the header always refers to the length of the data block in bytes.

For example, consider the following header:

#19<data>

#	Start of the header.
1	Number of decimal digits to follow to form the length.
9	Length of the data block (in bytes) that follows.
<data>	The pattern data, packed according the the DATA:PACKed command.

For non-packed data (or 8-bit packed data), the <block data> required to set an 8-bit pattern of alternating 1s and 0s (01010101) would be:

#11U (Note that "U" is the ASCII representation of 85)

For 4-bit packed data, the <block data> required to set the same pattern would be:

#1255

For 1-bit packed data, the <block data> would be as follows:

#1801010101

[SOURce[1]]:PATTern:UPATtern<n>:IDATa[?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.SetDataBlock (IVI-compliant)

Syntax [SOURce[1]]:PATTern:UFILe:IDATa [A | B,] <start bit>, <length in bits>, <block data>

[SOURce[1]]:PATTern:UFILe:IDATa? [A|B,] <start bit>, <length in bits>

Return Range The query returns the selected bits of the standard (A) or alternate (B) pattern.

Description This command is used to set specific bits in a user pattern. It is similar to the :DATA command. The :IDATa command is a contraction of the phrase **I**ncremental **DAT**a and is used to download part of a user-defined pattern.

The parameters have the following meanings:

Parameter	Description
[A|B]	Defines for which pattern the data is to be set (A = standard pattern, B = alternate pattern). If the pattern file describes a standard pattern (:USE = STRaight), this parameter cannot be B.
<filename>	Name of the file being defined. If the file does not exist, it is created.
<start bit>	First bit to be overwritten (starting with 0).
<length_in_bits>	Number of bits to be overwritten.
<block data>	The data that describes the pattern (see *"[SOURce[1]]:PATTern:UFILe:DATA[?]" on page 81* for the description).

The use of the parameters can be best illustrated by an example. If we have an alternate 16-bit pattern of 0s only, and we want to set the last four bits to 1s, the IDATa command would appear as follows:

- If the data packing is 8:

  SOURce1:PATTern:UPAT1:IDATa B, <filename>, 12, 4, #11(&F0)
  (where (&F0) is replaced by the ASCII representation of the value)

- If the data packing is 4:

  SOURce1:PATTern:UPAT1:IDATa B, <filename>, 12, 4, #11F

- If the data packing is 1:

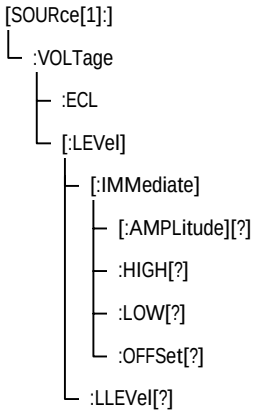
  SOURce1:PATTern:UPAT1:IDATa B, <filename>, 12, 4, #141111

The response form returns <block data> at the specified location.

NOTE See *"Working with User Patterns in SCPI" on page 47* for more information on using this command.

[SOURce[1]]:VOLTage Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:ECL	*"[SOURce[1]]:VOLTage:ECL" on page 90*
[:LEVel][:IMMediate][:AMPLitude][?]	*"[SOURce[1]]:VOLTage[:LEVel][:IMMediate][:AMPLitude][?]" on page 90*
[:LEVel][:IMMediate]:HIGH[?]	*"[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:HIGH[?]" on page 91*
[:LEVel][:IMMediate]:LOW[?]	*"[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:LOW[?]" on page 91*
[:LEVel][:IMMediate]:OFFSet[?]	*"[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:OFFSet[?]" on page 91*
[:LEVel]:LLEVel[?]	*"[SOURce[1]]:VOLTage[:LEVel]:LLEVel[?]" on page 91*

[SOURce[1]]:VOLTage:ECL

IVI-COM Equivalent IAgilentN490xPGOutput.LogicLevel (not IVI-compliant)

Syntax [SOURce[1]]:VOLTage:ECL

Description This command sets the data output values to those used for the ECL family. Retained for backwards compatibility. Superseded by SOURce1:VOLTage:LLEVel (see *"[SOURce[1]]:VOLTage[:LEVel]:LLEVel[?]" on page 91*).

[SOURce[1]]:VOLTage[:LEVel][:IMMediate][:AMPLitude][?]

IVI-COM Equivalent IAgilentN490xPGOutVoltage.VAmplitude (IVI-compliant)

Syntax [SOURce[1]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] <Num.>

[SOURce[1]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]?

Description The command sets the peak-to-peak value of the data signal in units of Volts. The query returns the peak-to-peak value of the data signal in units of Volts.

[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:HIGH[?]

IVI-COM Equivalent IAgilentN490xPGOutVoltage.VHigh (IVI-compliant)

Syntax [SOURce[1]]:VOLTage[:LEVel][:IMMediate]:HIGH <Num.>

[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:HIGH?

Description The command sets the DC low output level in units of Volts. The query returns the DC low output level in units of Volts.

[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:LOW[?]

IVI-COM Equivalent IAgilentN490xPGOutVoltage.VLow (IVI-compliant)

Syntax [SOURce[1]]:VOLTage[:LEVel][:IMMediate]:LOW <Num.>

[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:LOW?

Description The command sets the DC low output level in units of Volts. The query returns the DC low output level in units of Volts.

[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:OFFSet[?]

IVI-COM Equivalent IAgilentN490xPGOutVoltage.VOffset (IVI-compliant)

Syntax [SOURce[1]]:VOLTage[:LEVel][:IMMediate]:OFFSet <Num.>

[SOURce[1]]:VOLTage[:LEVel][:IMMediate]:OFFSet?

Description The command sets the mean of the high and low DC output level in units of Volts. The query returns the mean of the high and low DC output level in units of Volts.

[SOURce[1]]:VOLTage[:LEVel]:LLEVel[?]

IVI-COM Equivalent IAgilentN490xPGOutput.LogicLevel (not IVI-compliant)

Syntax [SOURce[1]]:VOLTage[:LEVel]:LLEVel <Family>

[SOURce[1]]:VOLTage[:LEVel]:LLEVel?

Input Parameters **<Family>** ECL | LVPECL | SCFL | LVDS | CML | CUSTom

Return Range ECL | LVPECL | SCFL | LVDS | CML | CUST

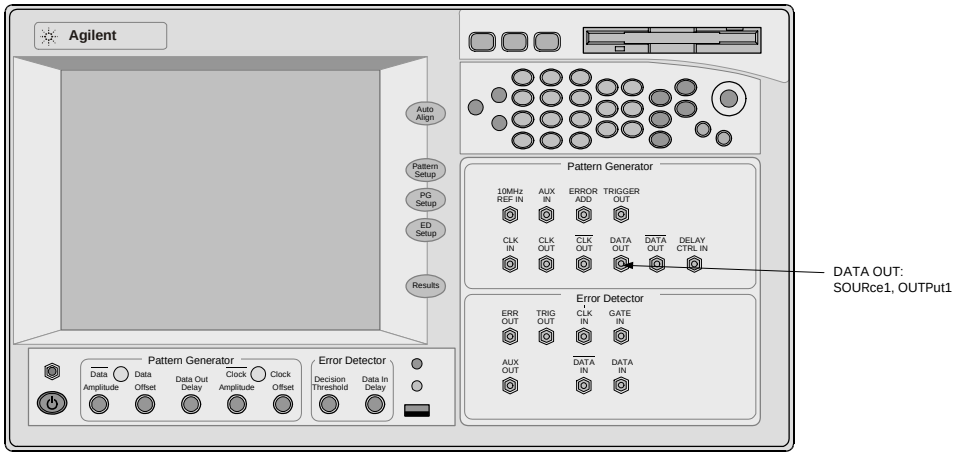
NOTE Selecting CUSTom has no effect.

Description The command sets the output level appropriate for the specified logic family. The query returns the currently used logic family.

NOTE If any of the voltage parameters have been modified, CUSTom will be returned by the query, even if the parameter has been set back to the default.

OUTPut[1] Subsystem

The Output[1] subsystem represents the pattern generator's Data Out port.



This subsystem has the following SCPI structure:

```
OUTPut[1]
  :CENTer
  :COUPling[?]
  :DATA
    :XOVer[?]
  :DELay[?]
  :POLarity[?]
  [:STATe][?]
  :TERMination[?]
```

This subsystem has the following commands:

Name	Description under
:CENTer	*"OUTPut[1]:CENTer" on page 94*
:COUPling[?]	*"OUTPut[1]:COUPling" on page 94*
:DATA:XOVer[?]	*"OUTPut[1]:DATA:XOVer[?]" on page 94*
:DELay[?]	*"OUTPut[1]:DELay[?]" on page 94*
:POLarity[?]	*"OUTPut[1]:POLarity[?]" on page 95*
[:STATe][?]	*"OUTPut[1][:STATe][?]" on page 95*
:TERMination[?]	*"OUTPut[1]:TERMination[?]" on page 95*

OUTPut[1]:CENTer

IVI-COM Equivalent IAgilentN490xPGGlobal.OutputsDisconnect (not IVI-compliant)

Syntax OUTPut[1]:CENTer DISConnect | CONNect

Description The DISConnect command sets the voltage at the pattern generator's Data Out port to 0 V, the CONNect command "reenables" the output (to the normal data pattern).

OUTPut[1]:COUPling

IVI-COM Equivalent IAgilentN490xPGOutput.TerminationEnabled (IVI-compliant)

Syntax OUTPut[1]:COUPling AC | DC

OUTPut[1]:COUPling?

Description The command enables or disables the source of the termination voltage:

- DC: Enables the termination voltage
- AC: Disables the termination voltage

The query returns the current state.

OUTPut[1]:DATA:XOVer[?]

IVI-COM Equivalent IAgilentN490xPGOutput.Crossover (IVI-compliant)

Syntax OUTPut[1]:DATA:XOVer <NR1.>

OUTPut[1]:DATA:XOVer? [MINimum | MAXimum]

The command sets the eye crossover of the pattern generator's Data Out port.

The query returns the current crossover.

OUTPut[1]:DELay[?]

IVI-COM Equivalent IAgilentN490xPGOutput.Delay (IVI-compliant)

Syntax OUTPut[1]:DELay <Num.>

OUTPut[1]:DELay?

Description This command sets the delay of the active edge of the clock output relative to the pattern generator's Data Out port. The units are seconds. The value is rounded to the nearest one picosecond. The response returns the current data to clock delay value.

This command has restrictions for frequencies under 620 Mbits/s. See the Serial BERT User Guide (or online Help) for details.

OUTPut[1]:POLarity[?]

IVI-COM Equivalent IAgilentN490xPGOutput.Polarity (IVI-compliant)

Syntax OUTPut[1]:POLarity NORMal | INVerted

OUTPut[1]:POLarity?

Return Range NORM | INV

Description The command sets the polarity of the pattern generator's Data Out port. The query returns the current polarity of the the pattern generator's Data Out port.

OUTPut[1][:STATe][?]

IVI-COM Equivalent IAgilentN490xPGOutput.Enabled (IVI-compliant)

Syntax OUTPut[1][:STATe] 0 | 1 | OFF | ON

OUTPut[1][:STATe]?

NOTE It is not possible to disable the Serial BERT's output. This command has no effect.

OUTPut[1]:TERMination[?]

IVI-COM Equivalent IAgilentN490xPGOutVoltage.VTermination (IVI-compliant)

Syntax OUTPut[1]:TERMination <NR3>

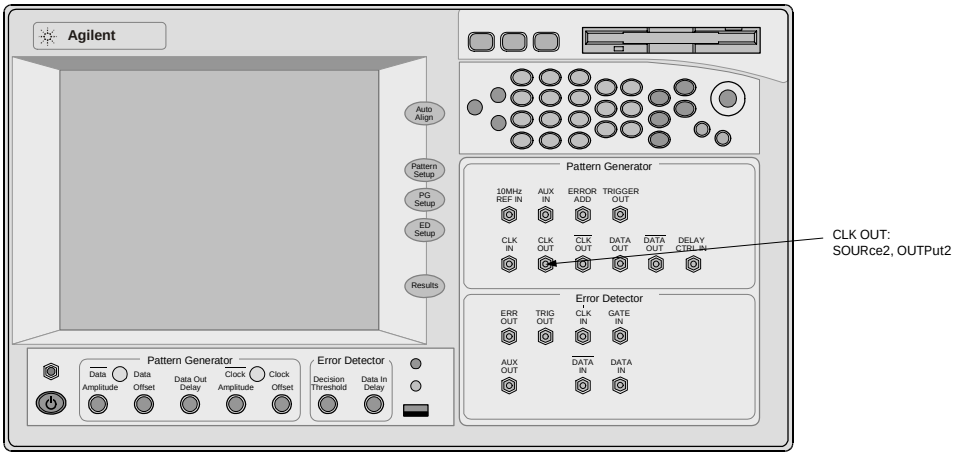
OUTPut[1]:TERMination?

Description This command sets the data termination level of the pattern generator's Data Out port. The response form returns the data termination level.

This command is only valid if the coupling is set to DC (see *"OUTPut[1]:COUPling" on page 94*).

SOURce9 Subsystem

The SOURce9 Subsystem represents the pattern generator's Clock Out port.



This subsystem has the following SCPI structure:

```
SOURce9
└ :FREQuency
  └ [:CW|:FIXed][?]
```

SOURce9:FREQuency[:CW|FIXed][?]

IVI-COM Equivalent IAgilentN490xPGClock.Frequency (IVI-compliant)

Syntax SOURce9:FREQuency[:CW|:FIXed] <Num.>

SOURce9:FREQuency[:CW|:FIXed]? <Num.>| <MIN | MAX>

Description This command may be used to configure the internal clock source frequency. You can also use any of the forms listed below:

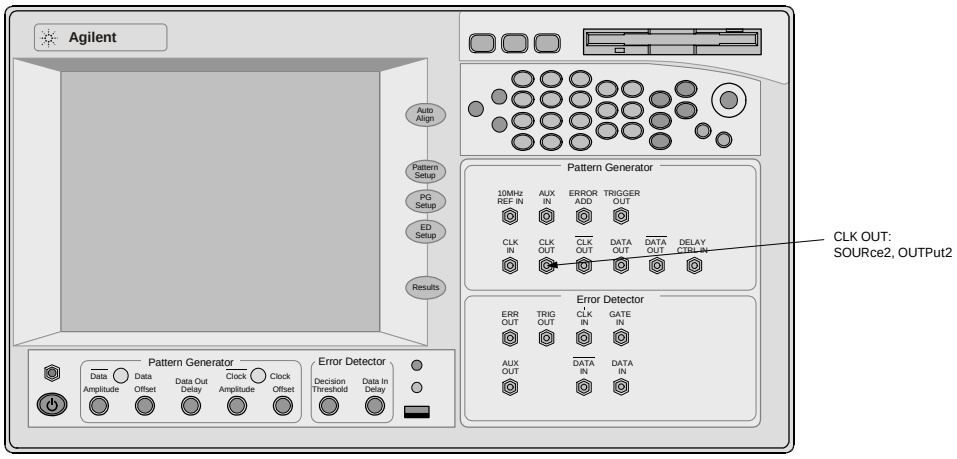
- SOURce9:FREQuency
- SOURce9:FREQuency:CW
- SOURce9:FREQuency:FIXed

There is no difference between any of these forms.

The response returns the current internal clock source frequency.

SOURce2 Subsystem

The SOURce2 Subsystem represents the pattern generator's Clock Out port.



This subsystem has the following SCPI structure:

```
SOURce2
├─ :FREQuency
│   └─ [:CW|:FIXed]?
└─ :VOLTage
    ├─ :ECL
    └─ [:LEVel]
        ├─ [:IMMediate]
        │   ├─ [:AMPLitude][?]
        │   ├─ :HIGH[?]
        │   ├─ :LOW[?]
        │   └─ :OFFSet[?]
        └─ :LLEVel
```

This subsystem has the following commands:

Name	Description under
:FREQuency[:CW|FIXed]?	*"SOURce2:FREQuency[:CW|:FIXed]?" on page 98*
:VOLTage:ECL	*"SOURce2:VOLTage:ECL" on page 98*
:VOLTage[:LEVEL][:IMMediate][:AMPLitude][?]	*"SOURce2:VOLTage[:LEVel][:IMMediate][:AMPLitude][?]" on page 99*
:VOLTage[:LEVEL][:IMMediate]:HIGH[?]	*"SOURce2:VOLTage[:LEVel][:IMMediate]:HIGH[?]" on page 99*
:VOLTage[:LEVEL][:IMMediate]:LOW[?]	*"SOURce2:VOLTage[:LEVel][:IMMediate]:LOW[?]" on page 99*
:VOLTage[:LEVEL][:IMMediate]:OFFSet[?]	*"SOURce2:VOLTage[:LEVel][:IMMediate]:OFFSet[?]" on page 99*
:VOLTage[:LEVEL]:LLEVel	*"SOURce2:VOLTage:LLEVel[?]" on page 100*

SOURce2:FREQuency[:CW|:FIXed]?

IVI-COM Equivalent IAgilentN490xPGClockIn.GetFrequency (IVI-compliant)

Syntax SOURce2:FREQuency[:CW|:FIXed]? [MIN|MAX]

Description This query returns the bit rate of the measured frequency from internal or external clock.

NOTE This query is superseded by SENSe6:FREQuency [:CW|:FIXed]?

SOURce2:VOLTage:ECL

IVI-COM Equivalent IAgilentN490xPGClock.LogicLevel (not IVI-compliant)

Syntax SOURce2:VOLTage:ECL

Description Sets the output AMPLitude and HIGH values to those used for the ECL family. There is no query form for this command. This command is provided for backwards compatibility only and is superseded by SOURce2:VOLTage:LLEVel (see *"SOURce2:VOLTage:LLEVel[?]" on page 100*).

SOURce2:VOLTage[:LEVel][:IMMediate][:AMPLitude][?]

IVI-COM Equivalent IAgilentN490xPGClockVoltage.VAmplitude (IVI-compliant)

Syntax SOURce2:VOLTage [:LEVel][:IMMediate][:AMPLitude] <Num.>

SOURce2:VOLTage [:LEVel][:IMMediate][:AMPLitude]?

Description The command sets the peak to peak value of the Clock Out signal in units of Volts. The query returns the peak to peak value of the Clock signal in units of Volts.

SOURce2:VOLTage[:LEVel][:IMMediate]:HIGH[?]

IVI-COM Equivalent IAgilentN490xPGClockVoltage.VHigh (IVI-compliant)

Syntax SOURce2:VOLTage[:LEVel][:IMMediate]:HIGH <Num.>

SOURce2:VOLTage[:LEVel][:IMMediate]:HIGH?

Description The command sets the DC high output level of the pattern generator's Clock Out port in Volts. The query returns the DC high output level of the pattern generator's Clock Out port in Volts.

SOURce2:VOLTage[:LEVel][:IMMediate]:LOW[?]

IVI-COM Equivalent IAgilentN490xPGClockVoltage.VLow (IVI-compliant)

Syntax SOURce2:VOLTage[:LEVel][:IMMediate]:LOW <Num.>

SOURce2:VOLTage[:LEVel][:IMMediate]:LOW?

The command sets the DC low output level of the pattern generator's Clock Out port in Volts. The query returns the DC low output level of the pattern generator's Clock Out port in Volts.

SOURce2:VOLTage[:LEVel][:IMMediate]:OFFSet[?]

IVI-COM Equivalent IAgilentN490xPGClockVoltage.VOffset (IVI-compliant)

Syntax SOURce2:VOLTage[:LEVel][:IMMediate]:OFFSet <Num.>

SOURce2:VOLTage[:LEVel][:IMMediate]:OFFSet?

Description The command sets the offset value of the pattern generator's Clock Out port in Volts. The query returns the offset value of the pattern generator's Clock Out port in Volts.

SOURce2:VOLTage:LLEVel[?]

IVI-COM Equivalent IAgilentN490xPGClockVoltage.LogicLevel (not IVI-compliant)

Syntax SOURce2:VOLTage:LLEVel <Family>

SOURce2:VOLTage:LLEVel?

Input Parameters **<Family>.** ECL | LVPECL | SCFL | LVDS | CML | CUSTom

NOTE Selecting CUSTom has no effect.

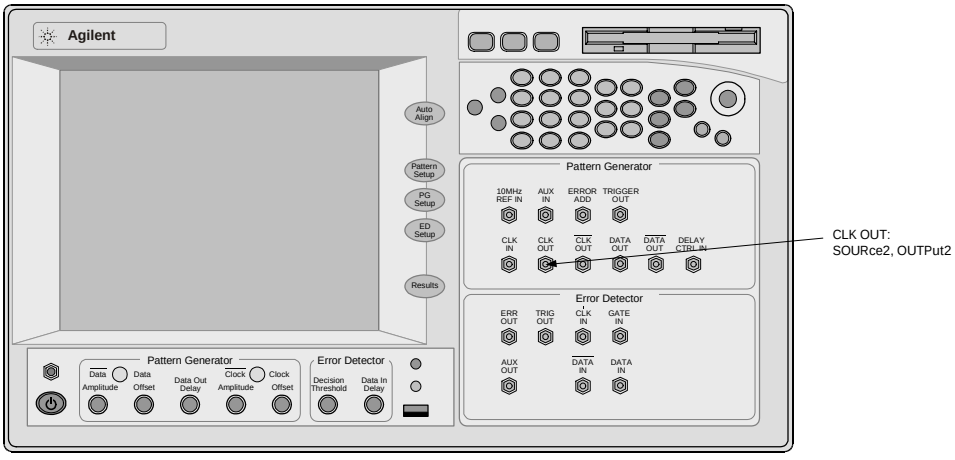
Return Range ECL | LVPECL | SCFL | LVDS | CML | CUSTom

Description The command sets the output level appropriate for the specified logic family. The query returns the currently used logic family.

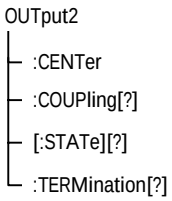
NOTE If any of the voltage parameters have been modified, CUSTom will be returned by the query, even if the parameter has been set back to the default.

OUTPut2 Subsystem

The OUTPut2 Subsystem represents the pattern generator's Clock Out port.



This subsystem has the following SCPI structure:



This subsystem has the following commands:

Name	Description under
:CENTer	*"OUTPut2:CENTer" on page 101*
:COUPling[?]	*"OUTPut2:COUPling[?]" on page 101*
[:STATe][?]	*"OUTPut2[:STATe][?]" on page 102*
:TERMination[?]	*"OUTPut2:TERMination[?]" on page 102*

OUTPut2:CENTer

IVI-COM Equivalent IAgilentN490xPGGlobal.OutputsDisconnect (not IVI-compliant)

Syntax OUTPut2:CENTer DISConnect | CONNect

Description The DISConnect command sets the voltage at the pattern generator's Data Out port to 0 V, the CONNect command "reenables" the output (to the normal data pattern).

OUTPut2:COUPling[?]

Syntax OUTPut2:COUPling AC | DC

OUTPut2:COUPling?

Description The command enables or disables the source of the termination voltage:

- DC: Enables the termination voltage
- AC: Disables the termination voltage

The query returns the current state.

OUTPut2[:STATe][?]

Syntax OUTPut2[:STATe] 0 | 1 | OFF | ON

OUTPut2[:STATe]?

Description The command controls the the pattern generator's Clock Out port. When OFF, the output is set to 0 V. The query returns the current state of the Clock Out (0 = OFF, 1 = ON).

NOTE It is not possible to disable the Serial BERT's output. This command has no effect.

OUTPut2:TERMination[?]

Syntax OUTPut2:TERMination 0 | –2 | 1.3

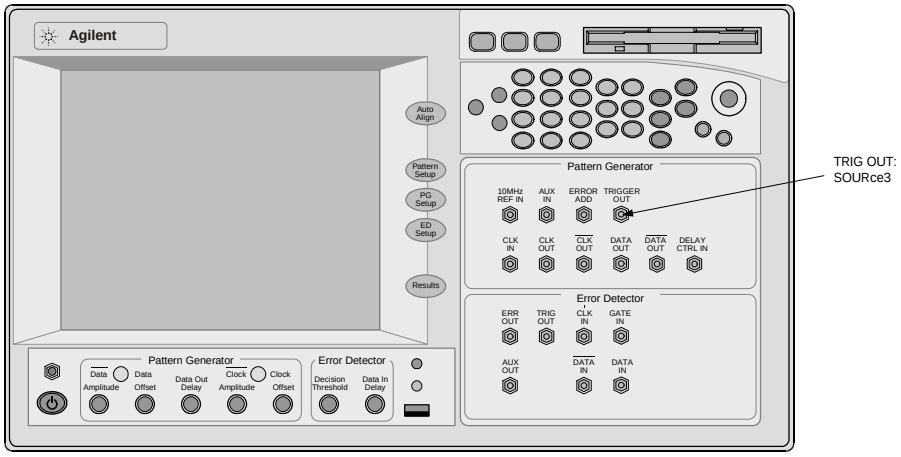
OUTPut2:TERMination?

Description This command sets the data termination level of the pattern generator's Clock Out port. The response form returns the data termination level.

This command is only valid if the coupling is set to DC (see *"OUTPut2:COUPling[?]" on page 101*).

SOURce3 Subsystem

The SOURce3 Subsystem represents the pattern generator's Trigger Out port.



This subsystem has the following SCPI structure:

```
SOURce3
└ :TRIGger
  ├ [:MODE][?]
  ├ :DCDRatio
  ├ :CTDRatio?
  ├ :APATtern<n>[?]
  ├ :MDENsity<n>[?]
  ├ :ZSUBstitut<n>[?]
  ├ :PRBN<n>[?]
  ├ :PRBS<n>[?]
  └ :UPATtern<n>
```

This subsystem has the following commands:

Name	Description under
:TRIGger[:MODe][?]	*"SOURce3:TRIGger[:MODe][?]" on page 104*
:TRIGger:DCDRatio	*"SOURce3:TRIGger:DCDRatio" on page 104*
:TRIGger:CTDRatio	*"SOURce3:TRIGger:CTDRatio?" on page 105*
:TRIGger:APATtern<n>[?]	*"SOURce3:TRIGger:APATtern<n>[?]" on page 105*
:TRIGger:MDENsity<n>[?]	*"SOURce3:TRIGger:MDENsity<n>[?]" on page 106*
:TRIGger:ZSUBstitut<n>[?]	*"SOURce3:TRIGger:ZSUBstitut<n>[?]" on page 106*
:TRIGger:PRBN<n>[?]	*"SOURce3:TRIGger:PRBN<n>[?]" on page 106*
:TRIGger:PRBS<n>[?]	*"SOURce3:TRIGger:PRBS<n>[?]" on page 107*
:TRIGger:UPATtern<n>	*"SOURce3:TRIGger:UPATtern<n>" on page 107*

NOTE See *"How Serial BERT Sends Triggers" on page 42* for details about trigger signals are generated.

SOURce3:TRIGger[:MODe][?]

IVI-COM Equivalent IAgilentN490xPGTrigger.Mode (IVI-compliant)

Syntax SOURce3:TRIGger[:MODE] DCLock | PATTern

SOURce3:TRIGger[:MODE]?

Return Range DCL | PATT

Description The command sets the pattern generator's Trigger Out to pattern or divided clock mode. The query returns the pattern generator's current Trigger Out mode.

SOURce3:TRIGger:DCDRatio

IVI-COM Equivalent IAgilentN490xPGTrigger.DivisionRate (IVI-compliant)

Syntax SOURce3:TRIGger:DCDRatio

Description The command sets the trigger subratio. CTDRatio? is the equivalent query.

SOURce3:TRIGger:CTDRatio?

IVI-COM Equivalent IAgilentN490xPGTrigger.DivisionRate (IVI-compliant)

Syntax SOURce3:TRIGger:CTDRatio

Description This query returns the trigger subratio. DCDRatio is the equivalent command.

SOURce3:TRIGger:APATtern<n>[?]

IVI-COM Equivalent IAgilentN490xPGTrigger.Patterntype (not IVI-compliant)

Syntax SOURce3:TRIGger:APATtern<n> ABCHange | SOPattern

SOURce3:TRIGger:APATtern<n>?

NOTE This command is for alternate patterns only.

Return Range ABCH | SOP

Description This command defines when a trigger should be sent from the pattern generator's Trigger Out port:

ABChange: The trigger is sent when the pattern being sent changes (from pattern A to pattern B or vice versa).

SOPattern: The pattern generator Trigger Out is synchronized to the start of a pattern.

The query returns the current state of the alternate pattern trigger mode.

NOTE See *"How the Serial BERT Uses Alternate Patterns" on page 40* for additional information on how to work with alternate patterns.

SOURce3:TRIGger:MDENsity<n>[?]

IVI-COM Equivalent IAgilentN490xPGPosition.Bit (not IVI-compliant)

Syntax SOURce3:TRIGger:MDENsity<n> <Num.>

SOURce3:TRIGger:MDENsity<n>?

Description This command selects the bit position within the PRBS at which the trigger pulse is to be output for MDEN patterns. The number <n> must be in the range: 7, 10, 11, 13, 15, 23. The parameter <Num> must be in the range 0 through pattern length – 1.

The query returns the bit position within the pattern at which the trigger pulse is to be output.

SOURce3:TRIGger:ZSUBstitut<n>[?]

IVI-COM Equivalent IAgilentN490xPGPosition.Bit (not IVI-compliant)

Syntax SOURce3:TRIGger:ZSUBstitut<n> <Num.>

SOURce3:TRIGger:ZSUBstitut<n>?

Description This command selects the bit position within the zero substituted 2^n PRBS at which the trigger pulse is to be output for ZSUB patterns. The number <n> must be in the range: 7, 10, 11, 13, 15, 23. The parameter <Num> must be in the range 0 through pattern length – 1.

The query returns the bit position within the pattern at which the trigger pulse is to be output.

SOURce3:TRIGger:PRBN<n>[?]

IVI-COM Equivalent IAgilentN490xPGPosition.Bit (not IVI-compliant)

Syntax SOURce3:TRIGger:PRBN<n> <Num.>

SOURce3:TRIGger:PRBN<n>?

Description This command selects the bit position within the PRBS at which the trigger pulse is to be output for PRBN patterns. The number <n> must be in the range: 7, 10, 11, 13, 15, 23. The parameter <Num> must be in the range 0 through pattern length – 1.

The query returns the bit position within the pattern at which the trigger pulse is to be output.

SOURce3:TRIGger:PRBS<n>[?]

IVI-COM Equivalent IAgilentN490xPGPosition.SetPattern (not IVI-compliant)

Syntax SOURce3:TRIGger:PRBS<n> <0 | 1 | OFF | ON>{,<0 | 1 | OFF | ON>}

SOURce3:TRIGger:PRBS<n>?

Description This command sets the pattern, the occurrence of which causes a trigger pulse to be output for PRBS patterns. In other words, when the defined pattern occurs, a trigger pulse is generated.

The number <n> must be in the range: 7, 10, 11, 15, 23, 31. The number of parameters depends on the pattern length, and is the minimum that can define a unique place in the overall pattern, for example a pattern of length 2^{n-1}, the number of parameters is n. The parameter values are either 1 or 0. An *all-ones* pattern is not allowed.

To generate a trigger pulse for a PRBS7 pattern on occurrence of 1010101, the following command would be sent:

```
SOUR3:TRIG:PRBS7 1,0,1,0,1,0,1
```

The query returns the state of the N-bit trigger pattern function for the pattern generator's Trigger Out.

SOURce3:TRIGger:UPATtern<n>

IVI-COM Equivalent IAgilentN490xPGPosition.Bit (not IVI-compliant)

Syntax SOURce3:TRIGger:UPATtern<n> <Num.>

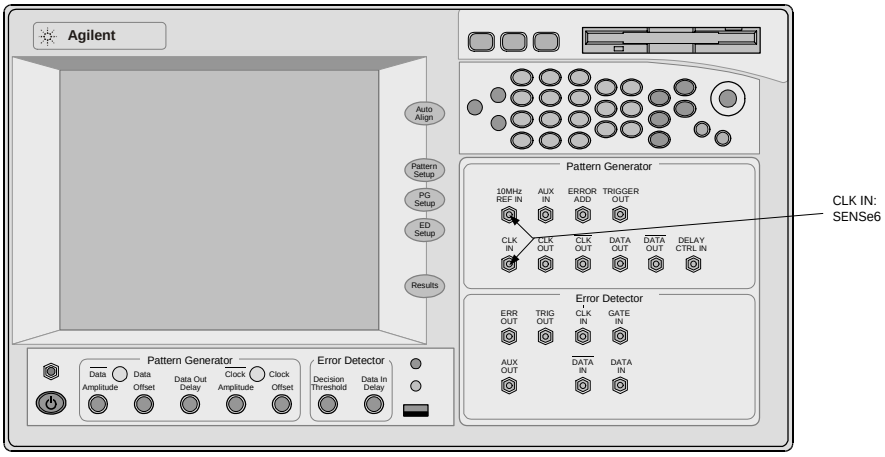
SOURce3:TRIGger:UPATtern<n>?

Description The command selects a bit position within the user pattern at which the trigger pulse is to be output for user patterns. The parameter must be in the range of 0 through pattern length – 1.

The response returns the current bit position within the user pattern at which the trigger pulse is generated.

SENSe6 Subsystem

The SENSe6 Subsystem represents the pattern generator's Clock In port.



This subsystem has the following SCPI structure:

```
SENSe6
├ :FREQuency
│  └ [:CW | :FIXed]
└ :MODe
```

This subsystem has the following commands:

Name	Description under
:FREQuency[:CW|:FIXed][?]	*"SENSe6:FREQuency[:CW|:FIXed]?" on page 109*
:MODe	*"SENSe6:MODe" on page 109*

SENSe6:FREQuency[:CW|:FIXed]?

IVI-COM Equivalent IAgilentN490xPGClockIn.GetFrequency (IVI-compliant)

Syntax SENSe6:FREQuency [:CW | :FIXed]?

Description This query returns the frequency of the signal at the pattern generator's Clock In port. You may also use the following forms of this query:

- SENSe6:FREQ?
- SENSe6:FREQ:CW?
- SENSe6:FREQ:FIXed?

There is no difference between any of these forms.

NOTE This command supersedes the following 716xxB command:

- SOURce2:FREQuency[:CW | :FIXed]? <Num.>

SENSe6:MODe

IVI-COM Equivalent IAgilentN490xPGClockIn.Mode (IVI-compliant)

Syntax SENSe6:MODe

Description This command sets the mode of the pattern generator's Clock In port. It can be one of the following:

- INTernal

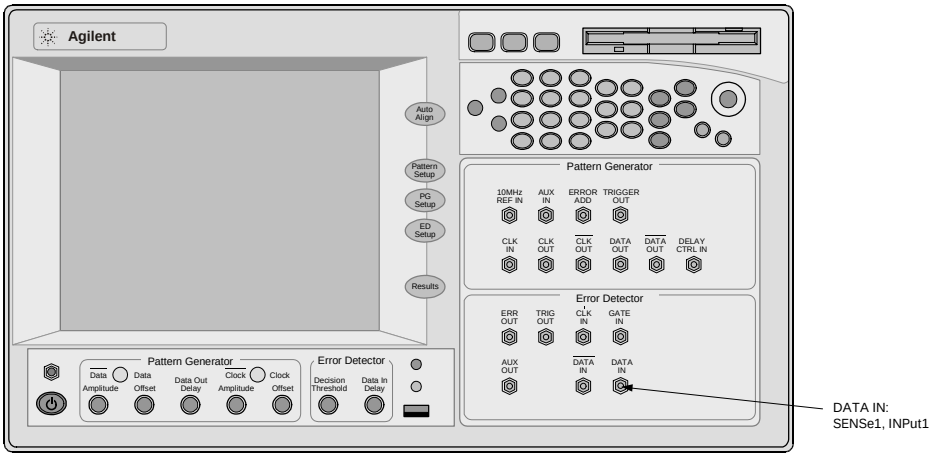
  This is the Serial BERT's internal clock.
- EXTernal

  This is the clock signal at the CLK IN port.
- REFerence (10-MHz Reference clock)

  This is the clock signal at the 10 MHz Ref port.

INPut[1] Subsystem

The INPut[1] subsystem represents the error detector's Data In port.



This subsystem has the following SCPI structure:

```
INPut[1]
 ├─ :COUPling[?]
 ├─ :DELay[?]
 ├─ :POLarity[?]
 ├─ :TERMination[?]
 ├─ :STATe[?]
 └─ :CMODe[?]
```

This subsystem has the following commands:

Name	Description under
:COUPling[?]	*"INPut[1]:COUPling[?]" on page 111*
:DELay[?]	*"INPut[1]:DELay[?]" on page 111*
:POLarity[?]	*"INPut[1]:POLarity[?]" on page 111*
:TERMination[?]	*"INPut[1]:TERMination[?]" on page 112*
:STATe[?]	*"INPut[1]:STATe[?]" on page 112*
:CMODe[?]	*"INPut[1]:CMODe[?]" on page 112*

INPut[1]:COUPling[?]

IVI-COM Equivalent IAgilentN490xEDDataIn.TerminationEnabled (IVI-compliant)

Syntax INPut[1]:COUPling AC | DC

INPut[1]:COUPling?

Description The command enables or disables the source of the termination voltage:

- DC: Enables the termination voltage
- AC: Disables the termination voltage

The query returns the current state.

NOTE Non-differential operation of the error detector's Data In port requires a termination voltage.

INPut[1]:DELay[?]

IVI-COM Equivalent IAgilentN490xEDDataIn.Delay (IVI-compliant)

Syntax INPut[1]:DELay <Num.>

INPut[1]:DELay?

Description This command sets the delay of the active edge of the clock output relative to the error detector's Data In port. The units are seconds. The value is rounded to the nearest one picosecond. The response returns the current data to clock delay value.

This command has restrictions for frequencies under 620 Mbits/s. See the Serial BERT User Guide (or online Help) for details.

INPut[1]:POLarity[?]

IVI-COM Equivalent IAgilentN490xEDDataIn.Polarity (IVI-compliant)

Syntax INPut[1]:POLarity NORMal|INVerted

INPut[1]:POLarity?

Description The command sets the polarity of the error detector's Data In port.

The query returns the current polarity of the the error detector's Data In port.

INPut[1]:TERMination[?]

IVI-COM Equivalent IAgilentN490xEDDataIn.VTermination (IVI-compliant)

Syntax INPut[1]:TERMination <NR3>

INPut[1]:TERMination?

Description This command sets the data termination level of the error detector's Data In port. The response form returns the data termination level.

This command is only valid if the coupling is set to DC (see *"INPut[1]:COUPling[?]" on page 111*).

If input termination and 0/1 threshold level are to be set up, the input termination should be set up first.

INPut[1]:STATe[?]

IVI-COM Equivalent IAgilentN490xEDDataIn.Enabled (IVI-compliant)

Syntax INPut[1]:STATe ON | OFF | 0 | 1

Description Enables or disables normal data input.

NOTE It is not possible to disable the Serial BERT's input. This command has no effect.

INPut[1]:CMODe[?]

IVI-COM Equivalent IAgilentN490xEDDataIn.Mode (IVI-compliant)

Syntax INPut[1]:CMODe DIFFerential | NORMal | COMPlement

INPut[1]:CMODe?

Description Defines the **C**ompare **MOD**e; this defines which input ports (DATA or $\overline{\text{DATA}}$) are active.

The following options are available:

- DIFFerential

  The differential between DATA and $\overline{\text{DATA}}$ is measured.

- NORMal

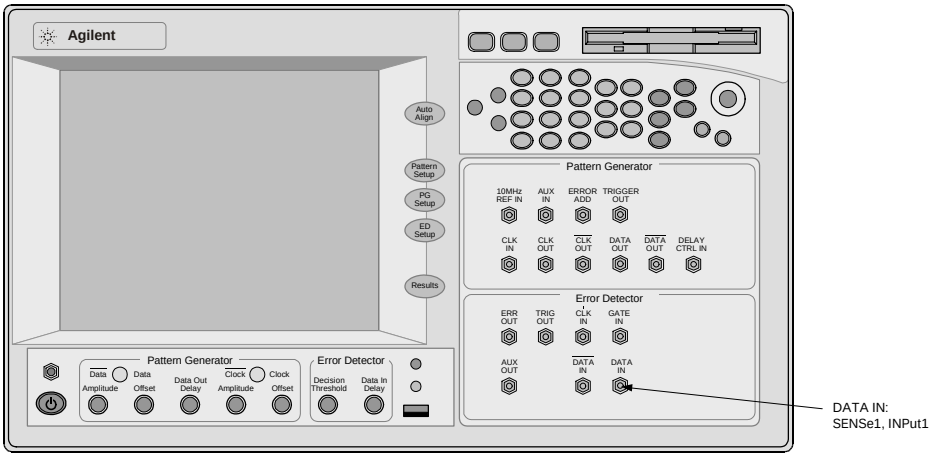
  Only the inputs at the DATA are measured.

- COMPlement

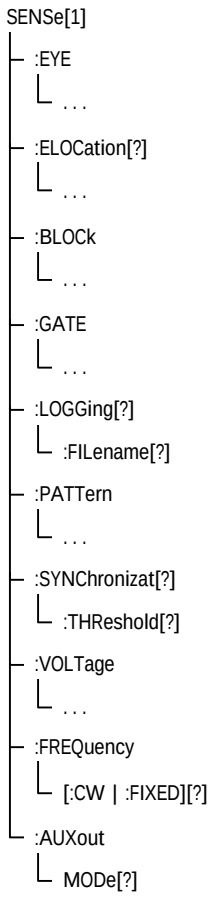
  Only the inputs at the $\overline{\text{DATA}}$ are measured.

SENSe[1] Subsystem

The SENSe[1] subsystem represents the error detector's Data In port.



This subsystem has the following SCPI structure:



This subsystem has the following subnodes and commands:

Name	Description under
Commands	
:LOGGing[?]	*"SENSe[1]:LOGGing[?]" on page 115*
:LOGGing:FILename[?]	*"SENSe[1]:LOGGing:FILename[?]" on page 116*
:SYNChronization[?]	*"SENSe[1]:SYNChronizat[?]" on page 116*
:SYNChronization:THReshold[?]	*"SENSe[1]:SYNChronization:THReshold[?]" on page 116*
:FREQuency[:CW|:FIXed][?]	*"SENSe[1]:FREQuency[:CW|:FIXed][?]" on page 117*
:AUXout:MODe[1][?]	*"SENSe[1]:AUXout:MODe[?]" on page 117*
Subnodes	
:BLOCk	*"SENSe[1]:BLOCk Subnode" on page 118*
:ELOCation	*"SENSe[1]:ELOCation Subnode" on page 121*
:EYE	*"SENSe[1]:EYE Subnode" on page 123*
:GATE	*"SENSe[1]:GATE Subnode" on page 129*
:PATTern	*"SENSe[1]:PATTern Subnode" on page 133*
:VOLTage	*"SENSe[1]:VOLTage Subnode" on page 146*

SENSe[1]:LOGGing[?]

IVI-COM Equivalent IAgilentN490xEDLogging.Enabled (IVI-compliant)

Syntax SENSe[1]:LOGGing 0 | 1 | OFF | ON

SENSe[1]:LOGGing?

Description This command allows you to save accumulated results in a file for later analysis. The query form returns whether or not the accumulated results will be saved in a log file.

NOTE This command is *not* precisely the same as 716xxB command. The ONCE parameter is no longer supported.

SENSe[1]:LOGGing:FILename[?]

IVI-COM Equivalent IAgilentN490xEDLogging.Filename (IVI-compliant)

Syntax SENSe[1]:LOGGing:FILename <Filename>

SENSe[1]:LOGGing:FILename?

Description This command specifies the file to which logged data is sent. The query returns the filename to which the logged data is sent.

You have to explicitly enable logging with SENSe[1]:LOGGing.

SENSe[1]:SYNChronizat[?]

IVI-COM Equivalent IAgilentN490xEDSynchronisation.AutoEnabled (IVI-compliant)

Syntax SENSe[1]:SYNChronizat ONCE | 0 | 1 | OFF | ON

SENSe[1]:SYNChronizat?

Description These commands configure the settings that control synchronization of the reference pattern to the incoming pattern.

- SENSe[1]:SYNChronizat ON enables automatic resynchronization.
- SENSe[1]:SYNChronizat OFF disables automatic resynchronization.
- SENSe[1]:SYNChronizat ONCE initiates a resynchronization attempt.

The query returns the current selection of the pattern synchronization.

SENSe[1]:SYNChronization:THReshold[?]

IVI-COM Equivalent IAgilentN490xEDSynchronisation.Threshold (IVI-compliant)

Syntax SENSe[1]:SYNChronization:THReshold

SENSe[1]:SYNChronization:THReshold?

Description SENSe[1]:SYNChronizat:THReshold <Num.>

SENSe[1]:SYNChronizat:THReshold?

This command sets the threshold level of error ratio at which synchronization is successful.

NOTE The valid values are 1E–01 thru 1E–08 in decade steps.

The query returns the threshold level of error ratio at which synchronization is set.

SENSe[1]:FREQuency[:CW|:FIXed][?]

IVI-COM Equivalent IAgilentN490xEDCDR.Frequency (not IVI-compliant)

Syntax SENSe[1]:FREQuency[:CW|:FIXed]

SENSe[1]:FREQuency[:CW|:FIXed]?

Description This command sets the clock frequency for clock data recovery. You can also use any of the forms listed below:

- SENSe[1]:FREQuency?
- SENSe[1]:FREQuency:CW?
- SENSe[1]:FREQuency:FIXed?

These forms have the same effect.

The query returns the clock frequency for clock data recovery.

SENSe[1]:AUXout:MODe[?]

IVI-COM Equivalent IAgilentN490xEDAuxOut.Mode (not IVI-compliant)

Syntax SENSe[1]:AUXout:MODe CLOCk | DATA

SENSe[1]:AUXout:MODe?

Description This command sets the mode for the error detector's Aux Out port. The following settings are available:

- CLOCk

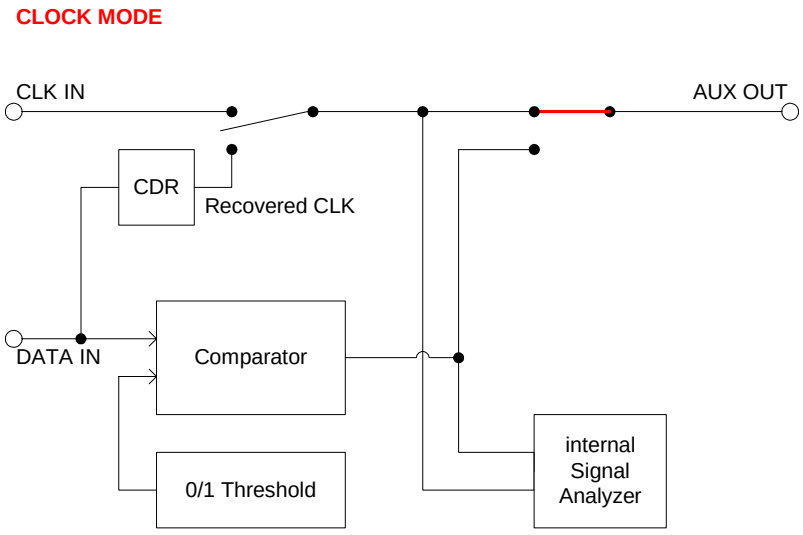
  The clock input is switched directly to the Aux Out port.

- DATA

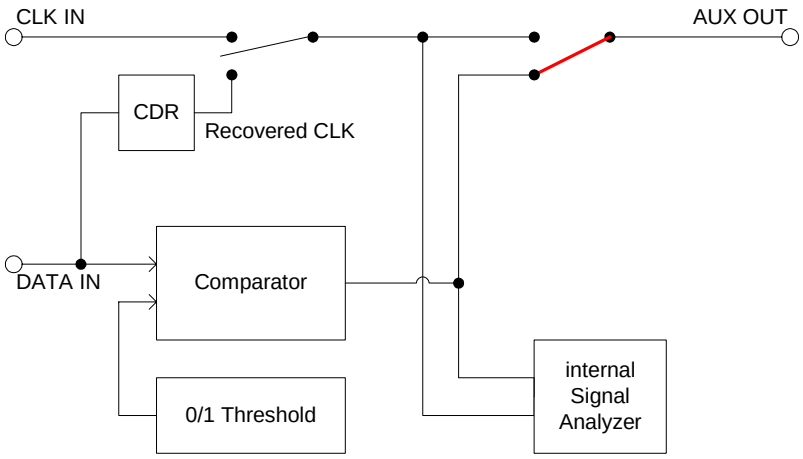
  The data input is switched via a comparator to the Aux Out port.

The comparator is controlled by the 0/1 threshold. You can use an oscilloscope to determine if the 0/1 threshold is correctly set. If the 0/1 threshold is set below or above the data eye, the output at Aux Out will be constant high or low, respectively.

The following figure shows how the clock signal is directed to Aux Out in CLOCk mode:



The following figure shows the circuit in DATA mode:



SENSe[1]:BLOCk Subnode

This subnode has the following SCPI structure:

```
SENSe[1]
└ :BLOCk[?]
  ├ BSTart[?]
  └ BLENgth[?]
```

This subnode has the following commands:

Name	Description under
:BLOCk[?]	*"SENSe[1]:BLOCk[?]" on page 119*
:BLOCk:BSTart[?]	*"SENSe[1]:BLOCk:BSTart[?]" on page 120*
:BLOCk:BLENgth[?]	*"SENSe[1]:BLOCk:BLENgth[?]" on page 120*

SENSe[1]:BLOCk[?]

IVI-COM Equivalent IAgilentN490xEDErrorLocation.Mode (not IVI-compliant)

Syntax SENSe[1]:BLOCk ON | OFF | 0 | 1 | BEADdress | WPATtern | BLOCk

SENSe[1]:BLOCk?

Description This command configures the error location feature of the instrument. It is only available for user-defined patterns (selected with PATTern:SELect UPATtern or PATTern:SELect FILename). Only the errors within the defined location are counted by the instrument.

Parameters The command has the following options:

- WPATtern

  When this is selected, all errors that occur through the entire pattern are counted. OFF and 0 (NULL) have the same effect.

- BEADdress

  Only errors that occur at the specified bit address are counted. Use :ELOC:BEAD to specify the bit address.

- BLOCk

  Errors that occur within the specified bit address range are counted. The bit address range is defined with the :BLOC:BST and BLOC:BLEN commands. ON and 1 have the same effect.

Return Values The query returns the following:

- 0 (equivalent to WPATtern)
- 1 (equivalent to BLOCk)
- BEAD

SENSe[1]:BLOCk:BSTart[?]

IVI-COM Equivalent IAgilentN490xEDErrorLocation.BlockStart (not IVI-compliant)

Syntax SENSe[1]:BLOCk:BSTart <numeric value>

SENSe[1]:BLOCk:BSTart?

Description Sets the starting bit address of a bit address range for error location. The query returns the current value. This command only has an effect if the BLOCk option is set with the SENSe[1]:BLOCk command.

This value must be in the range: 0 ... pattern length – 1. Values out of range are set to the maximum value silently.

SENSe[1]:BLOCk:BLENgth[?]

IVI-COM Equivalent IAgilentN490xEDErrorLocation.BlockLength (not IVI-compliant)

Syntax SENSe[1]:BLOCk:BLENgth <numeric value>

SENSe[1]:BLOCk:BLENgth?

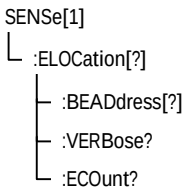
Description Sets the length of a bit address range for error location. The query returns the current value. This command only has an effect if the BLOCk option is set with the SENSe[1]:BLOCk command.

This value must be in the range: 1 ... (pattern length – one). Zero is set to 1 silently. Values beyond the maximum value are set to the maximum value silently.

NOTE The length is interpreted *round cycled*.

SENSe[1]:ELOCation Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:ELOCation[?]	*"SENSe[1]:ELOCation[?]" on page 121*
:ELOCation:BEADdress	*"SENSe[1]:ELOCation:BEADdress[?]" on page 122*
:ELOCation:VERBose?	*"SENSe[1]:ELOCation:VERBose?" on page 122*
:ELOCation:ECOunt?	*"SENSe[1]:ELOCation:ECOunt?" on page 123*

See the Serial BERT User's Guide and *"Using Error Location Capture – Procedures" on page 35* for additional information.

SENSe[1]:ELOCation[?]

IVI-COM Equivalent IAgilentN490xEDErrorLocation.CaptureErrors (not IVI-compliant)

Syntax SENSe[1]:ELOCation ONCE|OFF

SENSe[1]:ELOCation?

Description ONCE initiates the error location capture measurement. OFF stops a running error location measurement.

When an errored bit has been found, this command sets the location of the errored bit as the value that can be queried with SENSe[1]:ELOCation:BEADress?

The query returns the Error Location Capture *command* status. If 1 is returned, Error Location Capture has been triggered (but is not necessarily running). If 0 is returned, Error Location Capture has been either aborted (and may be still running) or successfully finished.

This is an overlapped command.

SENSe[1]:ELOCation:BEADdress[?]

IVI-COM Equivalent IAgilentN490xEDErrorLocation.BitAddress (not IVI-compliant)

Syntax SENSe[1]:ELOCation:BEADdress <numeric value>

SENSe[1]:ELOCation:BEADdress?

Description This command sets the bit address of the single bit that is to be monitored for errors. Only errors that occur at this bit are counted.

The return value of the query depends on the error location capture status:

- If an error has been captured, the bit position of the errored bit is returned.
- If no error has been captured since the last start of error location capture, the last set BEADdress is returned.

This command only has effect if the BEADdress option is selected with *"SENSe[1]:BLOCk[?]" on page 119*.

SENSe[1]:ELOCation:VERBose?

IVI-COM Equivalent IAgilentN490xEDErrorLocation.ReadState (not IVI-compliant)

Syntax SENSe[1]:ELOCation:VERBose?

Description This query returns the current state of the error location capture measurement. The following responses are possible:

- `ELOC__NOT_STARTED_YET`

  Status: Stopped; ELOC not started since last instrument power-up.

- `ELOC__RUNNING`

  Status: Running; ELOC is running, searching for an errored bit.

- `ELOC__ERROR_DETECTED`

  Status: Running; ELOC has detected ein errored bit. The data is being evaluated.

- `ELOC__SUCCESS`

  Status: Stopped; Last ELOC run was ended successfully: an errored bit was found in the bit stream and its bit position was calculated. The bit position is returned with ELOC:BEAD?.

- `ELOC__FAILED`

  Status: Stopped; last ELOC run was cancelled either due to a user error or internal error.

- ELOC__ABORTED

  Status: Stopped; last ELOC run was cancelled by the user (either from the user interface or remotely).

NOTE Please note that the responses each have two underscores after ELOC.

SENSe[1]:ELOCation:ECOunt?

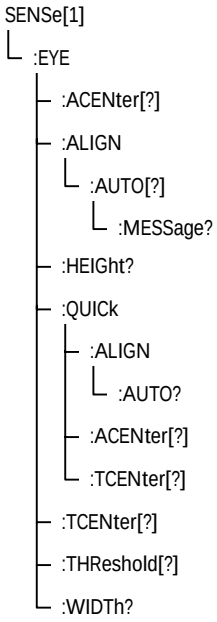
IVI-COM Equivalent IAgilentN490xEDErrorLocation.ReadCount (not IVI-compliant)

Syntax SENSe[1]:ELOCation:ECOunt?

Description Returns the number of bit-errors detected within the captured pattern. Is only set when an ELOC run was successful.

SENSe[1]:EYE Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:ACENter[?]	*"SENSe[1]:EYE:ACENter[?]" on page 124*
:ALIGN:AUTo[?]	*"SENSe[1]:EYE:ALIGN:AUTo[?]" on page 125*
:ALIGN:AUTo:MESSage?	*"SENSe[1]:EYE:ALIGN:AUTo:MESSage?" on page 126*
:HEIGht?	*"SENSe[1]:EYE:HEIGht?" on page 126*
:QUICk:ALIGn:AUTo?	*"SENSe[1]:EYE:QUICk:ALIGn:AUTo?" on page 126*
:QUICk:ACENter[?]	*"SENSe[1]:EYE:QUICk:ACENter[?]" on page 126*
:QUICk:TCENter[?]	*"SENSe[1]:EYE:QUICk:TCENter[?]" on page 127*
:TCENter[?]	*"SENSe[1]:EYE:TCENter[?]" on page 127*
:THReshold[?]	*"SENSe[1]:EYE:THReshold[?]" on page 128*
:WIDTh?	*"SENSe[1]:EYE:WIDTh?" on page 128*

SENSe[1]:EYE:ACENter[?]

IVI-COM Equivalent IAgilentN490xEDSampling.ZeroOneThresholdCenter (IVI-compliant)

Syntax SENSe[1]:EYE:ACENter

SENSe[1]:EYE:ACENter?

Description This command initiates a search for the 0/1 threshold voltage midway between the two 0/1 threshold voltages with a measured BER just in excess of the BER configured by the EYE:THReshold command. If successful, the command leaves the 0/1 threshold at this value, and the center of the eye can be found by querying the 0/1 threshold value.

If unsuccessful, EYE:HEIGht? returns 9.91E+37 (Not-A-Number, NAN). The command :ACENter OFF aborts a previously started search. When this command is in execution, SENSe[1]:VOLTage:ZOTHreshold:AUTo is set to OFF. The query returns the current value of the 0/1 threshold voltage.

This command is blocked for frequencies under 620 Mbits/s. See the Serial BERT User Guide (or online Help) for details.

NOTE This command temporarily disables autoalign.

NOTE The command :ACENter is an overlapped command. For more information, see *"Overlapped and Sequential Commands" on page 52*.

SENSe[1]:EYE:ALIGN:AUTo[?]

IVI-COM Equivalent IAgilentN490xEDSampling.AutoAlign (IVI-compliant)

Syntax SENSe[1]:EYE:ALIGN:AUTo ONCE | 0 | 1 | OFF | ON

SENSe[1]:EYE:ALIGN:AUTo?

Description This command is only available for instrument setups that include the following:

- BER Threshold <= 1.0E-2
- PRBS patterns (2^n-1)

This command enables/disables autoalign.

This command is blocked for frequencies under 620 Mbits/s. See the Serial BERT User Guide (or online Help) for details.

Return Parameters The query returns one of the following:

- CS_AUTO_ALIGN_INPROGRESS

  Auto Alignment in progress
- CS_CLOCKTODATA_ALIGN_INPROGRESS

  Clock to Data Alignment in progress
- CS_01THRESHOLD_INPROGRESS

  Threshold Alignment in progress
- CS_ABORTED

  Alignment interrupted by user command (*RST or :SENS1:EYE:ALIG:AUTo 0)
- CS_FAILED

  Alignment failed, the reason can be requested by *"SENSe[1]:EYE:ALIGN:AUTo:MESSage?" on page 126*.
- CS_SUCCESSFUL

  Alignment completed successfully, the eye parameters can be requested.

SENSe[1]:EYE:ALIGN:AUTo:MESSage?

IVI-COM Equivalent IIviDriverUtility.ErrorQuery (IVI-compliant)

Syntax SENSe[1]:EYE:ALIGN:AUTo:MESSage?

Description This query returns any message generated by the last autoalign. The message may be generated by autoalign, threshold center, or datadelay center functions. The message is returned as an unquoted string.

SENSe[1]:EYE:HEIGht?

IVI-COM Equivalent IAgilentN490xEDSampling.ReadEyeHeight (IVI-compliant)

Syntax SENSe[1]:EYE:HEIGht?

Description This is a query command that searches for the value of data amplitude that puts the 0/1 threshold level midway between the upper and lower bounds at which the error ratio exceeds the threshold value set by the :EYE:THReshold command.

If the result is not available or the search was unsuccessful, then the number 9.91E+37 (Not-A-Number, NAN) will be returned.

SENSe[1]:EYE:QUICk:ALIGn:AUTo?

Syntax SENSe[1]:EYE:QUICk:ALIGN:AUTo ONCE | 0 | 1 | OFF | ON

SENSe[1]:EYE:QUICk:ALIGN:AUTo? <Results>

Description This command calls *"SENSe[1]:EYE:ALIGN:AUTo[?]" on page 125*. It has no functionality of its own.

SENSe[1]:EYE:QUICk:ACENter[?]

Syntax SENSe[1]:EYE:QUICk:ACENter ONCE | 0 | 1 | OFF | ON

SENSe[1]:EYE:QUICk:ACENter?

Description This command calls *"SENSe[1]:EYE:ACENter[?]" on page 124*. It has no functionality of its own.

SENSe[1]:EYE:QUICk:TCENter[?]

Syntax SENSe[1]:EYE:QUICk:TCENter ONCE | 0 | 1 | OFF | ON

SENSe[1]:EYE:QUICk:TCENter?

Description This command calls *"SENSe[1]:EYE:TCENter[?]" on page 127*. It has no functionality of its own.

SENSe[1]:EYE:TCENter[?]

IVI-COM Equivalent IAgilentN490xEDSampling.ClockDataAlignCenter (IVI-compliant)

Syntax SENSe[1]:EYE:TCENter ONCE | 0 | 1 | OFF | ON

SENSe[1]:EYE:TCENter?

Description This command initiates a search for the value of data/clock delay that puts the active clock edge in the center of the data eye, midway between the two relative delay points with a measured BER just in excess of the BER configured by the EYE:THReshold command. If successful, the command leaves the data/clock delay at this value and the center of the eye can be found by querying the data delay value. If unsuccessful, EYE:WIDth? will return 9.91E+37 (Not-A-Number, NAN). The command :TCENter OFF aborts a previously started search.

NOTE The clock/data align feature (used to center the sampling point in the data input eye) uses information derived from the input clock frequency.

For the clock/data align feature to work properly, the input frequency must be stable during the measurement. The frequencies at the start and end of the measurement are compared, and if they differ by more than 10%, the measurement fails.

When a source clocking the instrument changes frequency, it will take time for the instrument to sense the change and adjust its configuration. Refer to the sections dealing with clock stabilization to ensure that the instrument's configuration has stabilized following any change of frequency prior to performing a clock to data alignment.

There is no need to alter the sync-mode before or after a clock to data alignment procedure, as AUTO sync-mode is automatically configured for the duration of the procedure.

NOTE The command :TCENter is an overlapped command. See *"Overlapped and Sequential Commands" on page 52*.

SENSe[1]:EYE:THReshold[?]

IVI-COM Equivalent IAgilentN490xEDSynchronisation.Threshold (IVI-compliant)

Syntax SENSe[1]:EYE:THReshold <Num.>

SENSe[1]:EYE:THReshold?

Description The command sets the BER threshold to be used in the determination of the edges of the eye.

The query returns the current BER threshold value.

SENSe[1]:EYE:WIDTh?

IVI-COM Equivalent IAgilentN490xEDSampling.ReadEyeWidth (IVI-compliant)

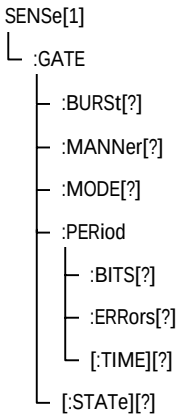
Syntax SENSe[1]:EYE:WIDTh?

Description This is a query that interrogates the eye width found by the most recent search for the value of data/clock delay that put the active edge in the center of the data eye.

If the result is not available or the search was unsuccessful, then 9.91E+37 (Not-A-Number, NAN) will be returned.

SENSe[1]:GATE Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:BURSt[?]	*"SENSe[1]:GATE:BURSt[?]" on page 129*
:MANNer[?]	*"SENSe[1]:GATE:MANNer[?]" on page 130*
:MODe[?]	*"SENSe[1]:GATE:MODe[?]" on page 130*
:PERiod:BITS[?]	*"SENSe[1]:GATE:PERiod:BITS[?]" on page 131*
:PERiod:ERRors[?]	*"SENSe[1]:GATE:PERiod:ERRors[?]" on page 131*
:PERiod[:TIMe][?]	*"SENSe[1]:GATE:PERiod[:TIMe][?]" on page 132*
[:STATe][?]	*"SENSe[1]:GATE[:STATe][?]" on page 132*

SENSe[1]:GATE:BURSt[?]

IVI-COM Equivalent IAgilentN490xEDAccumulation.BurstGatingEnabled (not IVI-compliant)

Syntax SENSe[1]:GATE:BURSt 0|1|OFF|ON

Description The command turns the burst sync mode OFF or ON.

The query returns the current setting.

SENSe[1]:GATE:MANNer[?]

IVI-COM Equivalent IAgilentN490xEDAccumulation.PeriodMode (not IVI-compliant)

Syntax SENSe[1]:GATE:MANNer <Manner>

SENSe[1]:GATE:MANNer?

Return Range TIME | ERR | BITS

Description This command sets the manner in which the accumulation period is controlled. <Manner> can be one of the following:

- TIME

  The error detector performs SINGle and REPetitive accumulation periods that are controlled by elapsed time.

- ERRors

  The error detector performs SINGle and REPetitive accumulation periods that are controlled by the accumulation of bit errors.

- BITS

  The error detector performs SINGle and REPetitive accumulation periods that are controlled by the accumulation of clock bits.

The query returns the current manner of accumulation.

SENSe[1]:GATE:MODe[?]

IVI-COM Equivalent IAgilentN490xEDAccumulation.ActivationMode (IVI-compliant)

Syntax SENSe[1]:GATE:MODE <Mode>

SENSe[1]:GATE:MODE?

Return Range MAN | SING | REP

Description The command sets the accumulation period mode **<Mode>**, which can be either:

- MANual

  The accumulation is started manually (by sending GATE:STATe ON).

- SINGle

  Errors are accumulated over one accumulation period.

- REPetitive

  The accumulation loops, according to the setting of :GATe:MANNer.

Executing this command invalidates all past results. The query returns the current accumulation mode.

SENSe[1]:GATE:PERiod:BITS[?]

IVI-COM Equivalent IAgilentN490xEDAccumulation.Bits (IVI-compliant)

Syntax SENSe[1]:GATE:PERiod:BITS <Num.>

SENSe[1]:GATE:PERiod:BITS?

Description When GATE:MANNer is set to BITS, the duration of the accumulation period is set in clock bits (or periods). **<Num.>** can be a value between 1E7 and 1E15 in decade steps.

Executing this command invalidates all past results.

The query returns the number of bits to which the gate period is set.

SENSe[1]:GATE:PERiod:ERRors[?]

IVI-COM Equivalent IAgilentN490xEDAccumulation.Errors (IVI-compliant)

Syntax SENSe[1]:GATE:PERiod:ERRors <Num.>

SENSe[1]:GATE:PERiod:ERRors?

Description When GATE:MANNer is set to ERRors, the command sets the duration of the accumulation period in bit errors. Values of 10, 100 and 1000 are permitted.

Executing this command invalidates all past results.

The query returns the number of errors to which the gate period is set.

SENSe[1]:GATE:PERiod[:TIMe][?]

IVI-COM Equivalent IAgilentN490xEDAccumulation.Time (IVI-compliant)

Syntax SENSe[1]:GATE:PERiod[:TIME] <Num.>

SENSe[1]:GATE:PERiod[:TIME]?

Description When GATE:MANNer is set to TIME, the duration of the accumulation period is set in seconds. Neither a value less than 1 second nor greater than 8639999 seconds (99 days, 23 hours, 59 minutes and 59 seconds) is permitted.

The command causes all past results to be labelled as invalid.

The query returns the time to which the gate period is set.

SENSe[1]:GATE[:STATe][?]

Syntax SENSe[1]:GATE[:STATe] 0 | 1 | OFF | ON

SENSe[1]:GATE[:STATe]?

This command turns accumulation on or off.

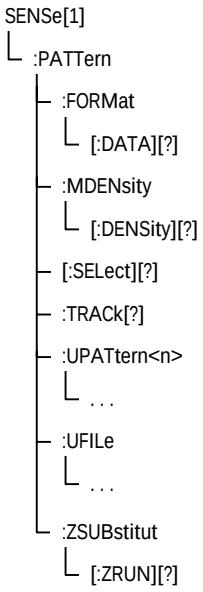
NOTE Previous commands that have altered the configuration of the instrument might not have settled. In order to ensure that the GATE ON command is not executed until conditions have settled, it is strongly recommended that the frequency be allowed to stabilize prior to the GATE ON command, and then be followed by a synchronization search. For more information, see *"When patterns are sent to the pattern generator or error detector, the Serial BERT requires some time to settle before. The following topics explain how the instruments react to pattern changes." on page 20*.

NOTE When GATE:MODE SINGle is executed, the GATE[:STATe]ON command is an overlapped command. For more information, see see *"Overlapped and Sequential Commands" on page 52*.

The query returns the current state of the accumulation gating.

SENSe[1]:PATTern Subnode

This subnode has the following SCPI structure:



This subnode has the following subnodes and commands:

Name	Description under
Commands	
:FORMat[:DATa][?]	*"SENSe[1]:PATTern:FORMat[:DATa][?]" on page 134*
:MDENsity[:DENSity][?]	*"SENSe[1]:PATTern:MDENsity[:DENSity][?]" on page 134*
[:SELect][?]	*"SENSe[1]:PATTern[:SELect][?]" on page 135*
:TRACk[?]	*"SENSe[1]:PATTern:TRACk[?]" on page 136*
:ZSUBstitut[:ZRUN][?]	*"SENSe[1]:PATTern:ZSUBstitut[:ZRUN][?]" on page 136*
Subnodes	
:UPATtern<n>	*"SENSe[1]:PATTern:UPATtern Subnode" on page 137*
:UFILe	*"SENSe[1]:PATTern:UFILe Subnode" on page 142*

SENSe[1]:PATTern:FORMat[:DATa][?]

-compliant)

Syntax SENSe[1]:PATTern:FORMat[:DATA] PACKed, <numeric value>

SENSe[1]:PATTern:FORMat[:DATA]?

Input Parameters **<PACKed>** permits the packing of bits within a byte to be set.

<NR1> Can be 1, 4, or 8.

Return Range 1 | 4 | 8

Description The command controls the format of data transfer for the :PATTern:UPATtern<n>:DATA, :PATTern:UPATtern<n>:IDATa, :PATTern:UFILe:DATA and :PATTern:UFILe:IDATa commands. The following values are possible:

- 1

  The data is sent as a string of 1s and 0s.

- 4

  The data is sent as a string of hex characters.

- 8

  The data is sent as a string of full ASCII characters.

The query returns the current value of the data pack.

See *"Working with User Patterns in SCPI" on page 47* for descriptions on how to use the data packing.

SENSe[1]:PATTern:MDENsity[:DENSity][?]

IVI-COM Equivalent IAgilentN490xEDDataIn.MarkDensity (not IVI-compliant)

Syntax SENSe[1]:PATTern:MDENsity[:DENSity] <numeric value>

SENSe[1]:PATTern:MDENsity[:DENSity]?

Input Parameters **<NR2>** 0.125, 0.25, 0.5, 0.75, 0.875

Description The command sets the ratio of high bits to the total number of bits in the pattern. The ratio may be varied in eighths, from one to seven (eighths), but excluding three and five.

The query returns the mark density in eighths.

SENSe[1]:PATTern[:SELect][?]

IVI-COM Equivalent IAgilentN490xEDDataIn.SelectData (IVI-compliant)

Syntax SENSe[1]:PATTern[:SELect] <Source>

SENSe[1]:PATTern[:SELect]?

Description This command defines the type of pattern being generated. The parameter is retained for backwards compatibility and may be one of the following:

PRBS<n>	<n> = 7, 10, 11, 15, 23, 31
PRBN<n>	<n> = 7, 10,11,13, 15, 23
ZSUBstitut<n>	<n> = 7, 10,11,13, 15, 23
UPATtern<n>	<n> = 1 through 12
MDENsity<n>	<n> = 7, 10,11,13, 15, 23
FILename,	<string>

ZSUBstitut **Z**ero **SUB**stitution; used for defining PRBN patterns in which a block of bits is replaced by a block of zeros. The length of the block is defined by *"SENSe[1]:PATTern:ZSUBstitut[:ZRUN][?]" on page 136*.

MDENsity **M**ark **DEN**sity; used for defining a PRBN pattern in which the user can set the mark density. The mark density is set with *"SENSe[1]:PATTern:MDENsity[:DENSity][?]" on page 134*.

UPATtern<n> **U**ser **PAT**tern; used to define the contents of a pattern store. For the Serial BERT, <n> can be 1 – 12.

FILename A parameter that allows the remote user to load a user pattern from the instrument's disk drive. This is the preferred mechanism for loading user patterns in the Serial BERT.

The query form returns the pattern's types in short form.

NOTE If a user-defined pattern is selected and the [:SELECT]? command is used, the response is UPAT. The particular value of <n> or the name of the file specified in the command form is not returned.

To get the path of a user pattern file, use the UFILe:NAME? command.

SENSe[1]:PATTern:TRACk[?]

IVI-COM Equivalent IAgilentN490xEDDataIn.TrackingEnabled (not IVI-compliant)

Syntax SENSe[1]:PATTern:TRACk 0 | 1 | OFF | ON

SENSe[1]:PATTern:TRACk?

Description This command enables and disables the error detector pattern tracking. When pattern tracking is enabled, the following commands sent to either the pattern generator or error detector are sent to both:

- SOURce[1] | SENSe[1] :PATTern:SELect[?]
- SOURce[1] | SENSe[1] :PATTern:FORMat[?]
- SOURce[1] | SENSe[1] :PATTern:MDENsity[:DENSity][?]
- SOURce[1] | SENSe[1] :PATTern:ZSUBstitut[:ZRUN][?]

The query returns the current state of the pattern track setting.

When pattern tracking is disabled, both instruments continue to use the current settings. Enabling pattern tracking causes the error detector to take over the settings of the pattern generator.

SENSe[1]:PATTern:ZSUBstitut[:ZRUN][?]

IVI-COM Equivalent IAgilentN490xEDDataIn.ZeroSub (not IVI-compliant)

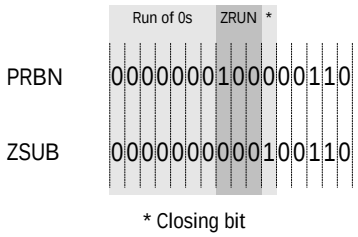
Syntax [SENSe[1]]:PATTern:ZSUBstitut[:ZRUN] MINimum | MAXimum | <numeric value>

[SENSe[1]]:PATTern:ZSUBstitut[:ZRUN]?

Return Range <NR3>

Description ZSUB patterns are PRBN patterns, where a number of bits are replaced by zeroes. The zero substitution starts after the longest runs of zeroes in the pattern (for example, for PRBN 2^7, after the run of 7 zeroes). This command allows you to define the length of the run of zeroes. For example, to produce 10 zeroes in a PRBN 2^7 pattern, three additional bits after the run of 7 zeroes must be replaced by zeroes. The bit after the run of zeroes (the closing bit) is set to 1.

The following figure shows an example, where a run of 10 zeroes is inserted into a PRBN 2^7 pattern.

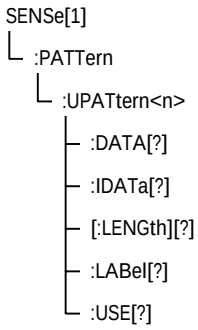


This command is only active when a ZSUB pattern has been selected (see *"SENSe[1]:PATTern[:SELect][?]" on page 135*).

Range The minimum value is the PRBN value – 1. The maximum value is length of the pattern – 1. So, for a PRBN 2^7 pattern, the minimum value is 6, and the maximum value is 127 (2^7 – 1).

SENSe[1]:PATTern:UPATtern Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:DATA[?]	*"SENSe[1]:PATTern:UPATtern<n>:DATA[?]" on page 138*
:IDATa[?]	*"SENSe[1]:PATTern:UPATtern<n>:IDATa[?]" on page 139*
[:LENGth][?]	*"SENSe[1]:PATTern:UPATtern<n>[:LENGth][?]" on page 140*
:LABel[?]	*"SENSe[1]:PATTern:UPATtern<n>:LABel[?]" on page 141*
:USE[?]	*"SENSe[1]:PATTern:UPATtern<n>:USE[?]" on page 141*

NOTE For the UPATtern<n> commands, <n> can be in the range 0 – 12. 0 (zero) is used to select the current pattern, 1 – 12 selects one of the user patterns in the memory.

SENSe[1]:PATTern:UPATtern<n>:DATA[?]

IVI-COM Equivalent IAgilentN490xEDPatternfile.SetData (IVI-compliant)

Syntax SENSe[1]:PATTern:UPATtern<n>:DATA [A|B,] <block_data>

SENSe[1]:PATTern:UPATtern<n>:DATA? [A|B]

Return Range The query returns the standard (A) or alternate pattern (B) of the file found under <filename>.

Description This command is used to set the bits in user pattern files. See *"Working with User Patterns in SCPI" on page 47* for a detailed description on how to edit user patterns.

The parameters have the following meanings:

Parameter	Description
[A|B]	Defines for which pattern the data is to be set (A = standard pattern, B = alternate pattern). If the pattern file describes a standard pattern (:USE = STRaight), this parameter cannot be B.
<filename>	Name of the file being defined. If the file does not exist, it is created.
<block data>	The data that describes the pattern (see the following for the description).

<block data> The <block data> parameter contains the actual data for setting the bits of the user pattern. The bits can also be packed using the FORMat[:DATA] command. If the bits are not packed, they are handled as 8-bit data. See *"[SOURce[1]]:PATTern:FORMat[:DATA][?]" on page 73*.

This command also sets the pattern length to fit the length of the data: If the data block is longer than the pattern, the pattern is extended to fit the data; if the data block is shorter than the pattern, the pattern is truncated to the end of the data.

<block data> starts with a header that indicates the length of the desired resulting data. The length of the <block data> embedded in the header always refers to the length of the data block in bytes.

For example, consider the following header:

#19<data>

#	Start of the header.
1	Number of decimal digits to follow to form the length.
9	Length of the data block (in bytes) that follows.
<data>	The pattern data, packed according the the DATA:PACKed command.

For non-packed data (or 8-bit packed data), the <block data> required to set an 8-bit pattern of alternating 1s and 0s (01010101) would be:

#11U (Note: "U" is the ASCII representation of binary 01010101.)

For 4-bit packed data, the <block data> required to set the same pattern would be:

#1255

For 1-bit packed data, the <block data> would be as follows:

#1801010101

SENSe[1]:PATTern:UPATtern<n>:IDATa[?]

IVI-COM Equivalent IAgilentN490xEDPatternfile.SetDataBlock (IVI-compliant)

Syntax SENSe[1]:PATTern:UPATtern<n>:IDATa [A|B,] <start_bit>, <length_in_bits>, <block_data>

SENSe[1]:PATTern:UPATtern<n>:IDATa? [A|B,] <start_bit>, <length_in_bits>

Return Range The query returns the selected bits of the standard (A) or alternate (B) pattern of the file found under <filename>.

Description This command is used to set specific bits in a user pattern. It is similar to the :DATA command. The :IDATa command is a contraction of the phrase **I**ncremental **DAT**a and is used to download part of a user-defined pattern.

The parameters have the following meanings:

Parameter	Description
[A|B]	Defines for which pattern the data is to be set (A = standard pattern, B = alternate pattern). If the pattern file describes a standard pattern (:USE = STRaight), this parameter cannot be B.
<filename>	Name of the file being defined. If the file does not exist, it is created.
<start bit>	First bit to be overwritten (starting with 0).
<length_in_bits>	Number of bits to be overwritten.
<block data>	The data that describes the pattern (see *"SENSe[1]:PATTern:UPATtern<n>:DATA[?]" on page 138* for the description).

The use of the parameters can be best illustrated by an example. If we have an alternate 16-bit pattern of 0s only, and we want to set the last four bits to 1s, the IDATa command would appear as follows:

- If the data packing is 8:

  SENSe[1]:PATTern:UPAT1:IDATa B, <filename>, 12, 4, #11(&F0) (where (&F0) is replaced by the ASCII representation of the value)

- If the data packing is 4:

  SENSe[1]:PATTern:UPAT1:IDATa B, <filename>, 12, 4, #11F

- If the data packing is 1:

  SENSe[1]:PATTern:UPAT1:IDATa B, <filename>, 12, 4, #141111

The response form returns <block data> at the specified location.

NOTE See *"Working with User Patterns in SCPI" on page 47* for more information on using this command.

SENSe[1]:PATTern:UPATtern<n>[:LENGth][?]

IVI-COM Equivalent IAgilentN490xEDPatternfile.Length (IVI-compliant)

Syntax SENSe[1]:PATTern:UPATtern<n>[:LENGth] <Num.>

SENSe[1]:PATTern:UPATtern<n>[:LENGth]?

Description This command sets the length of a user pattern file. The query returns the length of the user pattern file. If an alternate pattern is selected (:USE APATtern), the LENGth command sets the length of each half of the pattern.

Note that the :DATA command sets the length of the file.

See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

SENSe[1]:PATTern:UPATtern<n>:LABel[?]

IVI-COM Equivalent IAgilentN490xEDPatternfile.Description (IVI-compliant)

Syntax SENSe[1]:PATTern:UPATtern<n>:LABel <string>

SENSe[1]:PATTern:UPATtern<n>:LABel?

Description This command sets a description for a user pattern file. The query returns the description. See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

SENSe[1]:PATTern:UPATtern<n>:USE[?]

IVI-COM Equivalent IAgilentN490xEDPatternfile.Alternate (IVI-compliant)

Syntax SENSe[1]:PATTern:UPATtern<n>:USE STRaight | APATtern

SENSe[1]:PATTern:UPATtern<n>:USE?

Return Range STR | APAT

Description This command defines whether a user pattern file should be a straight pattern or an alternate pattern:

- STRaight

  The pattern is repeatedly output.

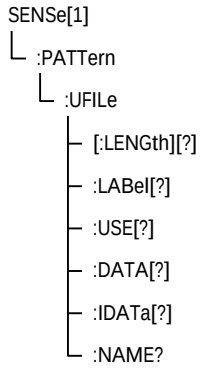
- APATtern

  The pattern is composed of two halves. The output depends on various other commands; see *"How the Serial BERT Uses Alternate Patterns" on page 40* for more information.

The default is set to have a length of 128 bits for each half pattern; all bits are set to zero and the trigger is set to occur on the A/B changeover. See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

SENSe[1]:PATTern:UFILe Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
[:LENGth][?]	*"SENSe[1]:PATTern:UFILe[:LENGth][?]" on page 142*
:LABel[?]	*"SENSe[1]:PATTern:UFILe:LABel[?]" on page 143*
:USE[?]	*"SENSe[1]:PATTern:UFILe:USE[?]" on page 143*
:DATa[?]	*"SENSe[1]:PATTern:UFILe:DATa[?]" on page 143*
:IDATa[?]	*"SENSe[1]:PATTern:UFILe:IDATa[?]" on page 145*
:NAMe[?]	*"SENSe[1]:PATTern:UFILe:NAMe[?]" on page 146*

SENSe[1]:PATTern:UFILe[:LENGth][?]

IVI-COM Equivalent IAgilentN490xLocalPatternfile.Length (IVI-compliant)

Syntax SENSe[1]:PATTern:UFILe[:LENGth] <filename>, <numeric_value>

SENSe[1]:PATTern:UFILe[:LENGth]? <filename>

Description This command sets the length of a user pattern file. The query returns the length of the user pattern file. If an alternate pattern is selected (:USE APATtern), the LENGth command sets the length of each half of the pattern.

Note that the :DATA command automatically sets the length of the file.

See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

SENSe[1]:PATTern:UFILe:LABel[?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.Description (IVI-compliant)

Syntax SENSe[1]:PATTern:UFILe:LABel <filename>, <string>

SENSe[1]:PATTern:UFILe:LABel? <filename>

Description This command sets a description for a user pattern file. The query returns the description. See *"Working with User Patterns in SCPI" on page 47* for information on using this command.

SENSe[1]:PATTern:UFILe:USE[?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.Alternate (IVI-compliant)

Syntax SENSe[1]:PATTern:UFILe:USE <filename>, STRaight | APATtern

SENSe[1]:PATTern:UFILe:USE? <filename>

Description This command defines whether a user pattern file should be a straight pattern or an alternate pattern:

- STRaight

  The pattern is repeatedly output.

- APATtern

  The pattern is composed of two halves. The output depends on various other commands; see *"How the Serial BERT Uses Alternate Patterns" on page 40* for more information.

The default is set to have a length of 128 bits for each half pattern; all bits are set to zero and the trigger is set to occur on the A/B changeover.

SENSe[1]:PATTern:UFILe:DATa[?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.SetData (IVI-compliant)

Syntax SENSe[1]:PATTern:UFILe:DATA [A | B,] <filename>, <block_data>

SENSe[1]:PATTern:UFILe:DATA? [A|B,] <filename>

Return Range The query returns the standard (A) or alternate pattern (B) of the file found under <filename>.

Description This command is used to set the bits in user pattern files. See *"Working with User Patterns in SCPI" on page 47* for a detailed description on how to edit user patterns.

The parameters have the following meanings:

Parameter	Description
[A|B]	Defines for which pattern the data is to be set (A = standard pattern, B = alternate pattern). If the pattern file describes a standard pattern (:USE = STRaight), this parameter cannot be B.
<filename>	Name of the file being defined. If the file does not exist, it is created.
<block data>	The data that describes the pattern (see the following for the description).

<block data> The <block data> parameter contains the actual data for setting the bits of the user pattern. The bits can also be packed using the FORMat[:DATA] command. If the bits are not packed, they are handled as 8-bit data. See *"[SOURce[1]]:PATTern:FORMat[:DATA][?]" on page 73*.

This command also sets the pattern length to fit the length of the data: If the data block is longer than the pattern, the pattern is extended to fit the data; if the data block is shorter than the pattern, the pattern is truncated to the end of the data.

<block data> starts with a header that indicates the length of the desired resulting data. The length of the <block data> embedded in the header always refers to the length of the data block in bytes.

For example, consider the following header:

#19<data>

#	Start of the header.
1	Number of decimal digits to follow to form the length.
9	Length of the data block (in bytes) that follows.
<data>	The pattern data, packed according the the DATA:PACKed command.

For non-packed data (or 8-bit packed data), the <block data> required to set an 8-bit pattern of alternating 1s and 0s (01010101) would be:

#11U (Note that "U" is the ASCII representation of 85)

For 4-bit packed data, the <block data> required to set the same pattern would be:

#1255

For 1-bit packed data, the <block data> would be as follows:

#1801010101

SENSe[1]:PATTern:UFILe:IDATa[?]

IVI-COM Equivalent IAgilentN490xPGPatternfile.SetDataBlock (IVI-compliant)

Syntax SENSe[1]:PATTern:UFILe:IDATa [A | B,] <filename>, <start_bit>, <length_in_bits>, <block_data>

SENSe[1]:PATTern:UFILe:IDATa? [A|B,] <filename>, <start_bit>, <length_in_bits>

Return Range The query returns the selected bits of the standard (A) or alternate (B) pattern of the file found under <filename>.

Description This command is used to set specific bits in a user pattern. It is similar to the :DATA command. The :IDATa command is a contraction of the phrase **I**ncremental **DAT**a and is used to download part of a user-defined pattern.

The parameters have the following meanings:

Parameter	Description
[A|B]	Defines for which pattern the data is to be set (A = standard pattern, B = alternate pattern). If the pattern file describes a standard pattern (:USE = STRaight), this parameter cannot be B.
<filename>	Name of the file being defined. If the file does not exist, it is created.
<start bit>	First bit to be overwritten (starting with 0).
<length_in_bits>	Number of bits to be overwritten.
<block data>	The data that describes the pattern (see *"[SOURce[1]]:PATTern:UFILe:DATA[?]" on page 81* for the description).

The use of the parameters can be best illustrated by an example. If we have an alternate 16-bit pattern of 0s only, and we want to set the last four bits to 1s, the IDATa command would appear as follows:

- If the data packing is 8:

  SOURce1:PATTern:UFILe:IDATa B, <filename>, 12, 4, #11(&F0)
  (where (&F0) is replaced by the ASCII representation of the value)

- If the data packing is 4:

  SOURce1:PATTern:UFILe:IDATa B, <filename>, 12, 4, #11F

- If the data packing is 1:

  SOURce1:PATTern:UFILe:IDATa B, <filename>, 12, 4, #141111

The response form returns <block data> at the specified location.

NOTE See *"Working with User Patterns in SCPI" on page 47* for more information on using this command.

SENSe[1]:PATTern:UFILe:NAMe[?]

Syntax SENSe[1]:PATTern:UFILe:NAME?

Description This query returns the file name of the currently used user pattern. It is only valid if SENSe[1]:PATTern:SELect? returns UPAT.

SENSe[1]:VOLTage Subnode

This subnode has the following SCPI structure:

This subnode has the following commands:

Name	Description under
:ZOTHreshold[?]	*"SENSe[1]:VOLTage:ZOTHreshold[?]" on page 147*
:ZOTHreshold:RANGe[:HIGH][?]	*"SENSe[1]:VOLTage:ZOTHreshold:RANGe[:HIGH][?]" on page 147*
:ZOTHreshold:RANGe:LOW[?]	*"SENSe[1]:VOLTage:ZOTHreshold:RANGe:LOW[?]" on page 147*
:ZOTHreshold:AUTo[?]	*"SENSe[1]:VOLTage:ZOTHreshold:AUTo[?]" on page 148*

SENSe[1]:VOLTage:ZOTHreshold[?]

IVI-COM Equivalent IAgilentN490xEDSampling.ZeroOneThreshold (IVI-compliant)

Syntax SENSe[1]:VOLTage:ZOTHreshold <Num.>

SENSe[1]:VOLTage:ZOTHreshold?

Description This command sets the level at which the error detector discriminates between a 0 and a 1.

A numeric value parameter sets the level to a given value in Volts. It also sets :ZOTHreshold:AUTo to OFF.

When in :ZOTHreshold:AUTo OFF, the query returns the last user-entered value.

When in :ZOTHreshold:AUTo ON, the query returns the value automatically determined by the hardware.

If you are going to use this command to set the 0/1 threshold, first disable the automatic mode with :ZOTHreshold:AUTo OFF.

SENSe[1]:VOLTage:ZOTHreshold:RANGe[:HIGH][?]

IVI-COM Equivalent IAgilentN490xEDSampling.ZeroOneThresholdVHigh (not IVI-compliant)

Syntax SENSe[1]:VOLTage:ZOTHreshold:RANGe[:HIGH] <NR3>

SENSe[1]:VOLTage:ZOTHreshold:RANGe[:HIGH]?

Description Sets/returns the higher limit of the input range of the zero/one threshold.

SENSe[1]:VOLTage:ZOTHreshold:RANGe:LOW[?]

IVI-COM Equivalent IAgilentN490xEDSampling.ZeroOneThresholdVLow (not IVI-compliant)

Syntax SENSe[1]:VOLTage:ZOTHreshold:RANGe:LOW <NR3>

SENSe[1]:VOLTage:ZOTHreshold:RANGe:LOW?

Description Sets/returns the lower limit of the input range of the zero/one threshold.

SENSe[1]:VOLTage:ZOTHreshold:AUTo[?]

IVI-COM Equivalent IAgilentN490xEDSampling.ZeroOneThresholdTrack (not IVI-compliant)

Syntax SENSe[1]:VOLTage:ZOTHreshold:AUTo 0 | 1 | OFF | ON

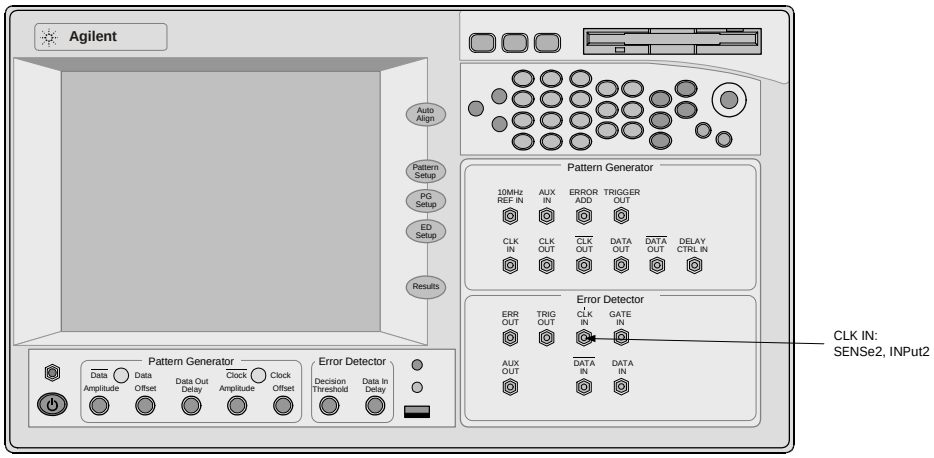
SENSe[1]:VOLTage:ZOTHreshold:AUTo?

Description This command enables an automatic mode, in which the 0/1 threshold level is set to the mean of the input signal.

The query returns the current setting of the hardware discrimination circuit.

INPut2 Subsystem

The INPut2 Subsystem represents the error detector's Clock In port.



This subsystem has the following SCPI structure:

```
INPut2
 ├ :TERMination[?]
 └ :COUPling[?]
```

INPut2:TERMination[?]

Syntax INPut2:TERMination 0 | –2 | 1.3

INPut2:TERMination?

This command is obsolete. It has no effect.

INPut2:COUPling[?]

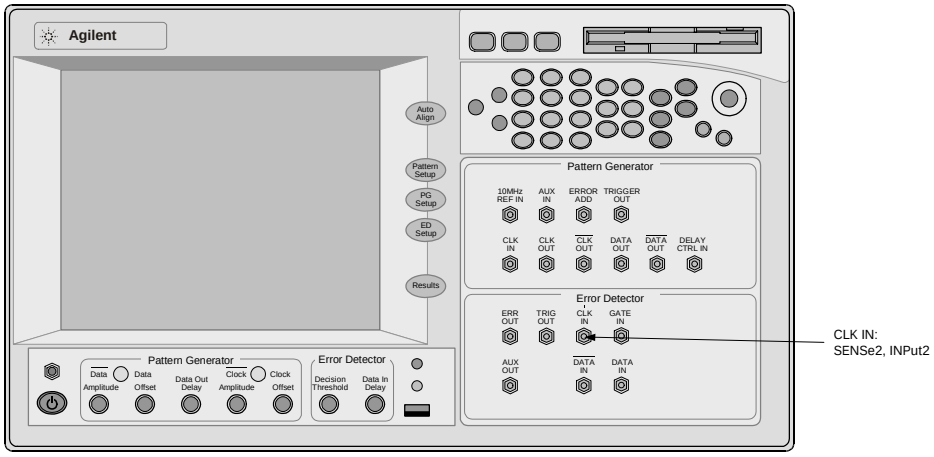
Syntax INPut2:COUPling AC | DC

INPut2:COUPling?

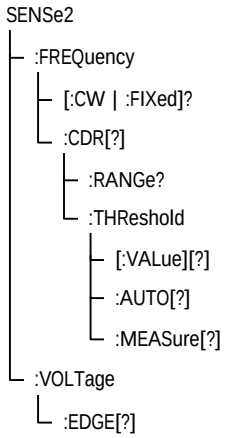
This command is obsolete. It has no effect.

SENSe2 Subsystem

The SENSe2 Subsystem controls the error detector's Clock In port.



This subsystem has the following SCPI structure:



This subsystem has the following commands:

Name	Description under
:FREQuency[:CW | FIXed]?	*"SENSe2:FREQuency[:CW|:FIXed]?" on page 150*
:FREQuency:CDR[?]	*"SENSe2:FREQuency:CDR[?]" on page 151*
:FREQuency:CDR:RANGe?	*"SENSe2:FREQuency:CDR:RANGe?" on page 152*
:FREQuency:CDR:AUTo[?]	*"SENSe2:FREQuency:CDR:AUTo[?]" on page 152*
:FREQuency:CDR:THReshold[:VALue][?]	*"SENSe2:FREQuency:CDR:THReshold[:VALue][?]" on page 153*
:FREQuency:CDR:THReshold:MEASure[?]	*"SENSe2:FREQuency:CDR:THReshold:MEASure[?]" on page 153*
:VOLTage:EDGe[?]	*"SENSe2:VOLTage:EDGe[?]" on page 153*

SENSe2:FREQuency[:CW|:FIXed]?

IVI-COM Equivalent IAgilentN490xEDClockIn.GetFrequency (IVI-compliant)

Syntax SENSe2:FREQuency[:CW|:FIXed]?

Description This query returns the internal clock source frequency. You can also use any of the forms listed below:

- SENSe2:FREQuency?
- SENSe2:FREQuency:CW?

- SENSe2:FREQuency:FIXed?

These forms have the same effect.

SENSe2:FREQuency:CDR[?]

IVI-COM Equivalent IAgilentN490xEDCDR.Enabled (not IVI-compliant)

Syntax SENSe2:FREQuency:CDR ON | OFF

SENSe2:FREQuency:CDR?

Description Enables or disables clock data recovery (CDR) mode.

How Does Clock Data Recovery Work? In CDR mode, the CDR has to recover the clock from the incoming data. To do this, the hardware has to decide whether the voltage at the input connector is a logical '1' or '0' and then recover the clock from the detected transitions.

Because the regular threshold voltage is not only used to determine the optimum sampling for the data, but also to perform measurements such as eye diagram or output level measurements, it is not possible to use it for the clock recovery.

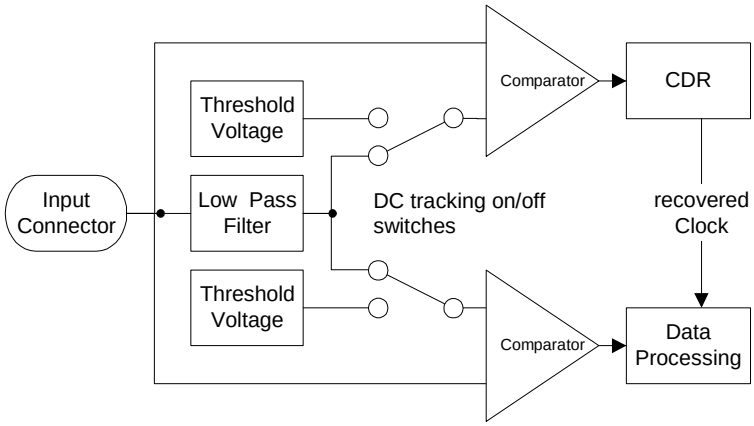
For this reason, the clock recovery circuitry has it's own comparator for the incoming data. This comparator also needs to know the threshold voltage (0/1 decision threshold).

The threshold voltage can be derived from the input signal via a low-pass filter. This will work fine for most applications. But applications that do not provide a continuous data stream at the input (for example, any application using bursts) cannot use this low-pass filter, because the threshold voltage will drift from the correct level when there is no input. In such cases, the threshold can be specified manually. It is then no longer derived from the input signal (see the following figure). The manually set threshold voltage must of course be within the input range.

The difference between the data path and the CDR path is that the comparator of the CDR is always single-ended. Thus, this comparator always needs a threshold voltage that lies between the high and low levels of the incoming signal.

The differential threshold of the data path comparator has no relation to the single-ended threshold of the CDR path comparator. This means that in differential mode, the two thresholds will be different and in single-ended mode (either normal and complement) they will/can be equal (except during measurements).

The following figure shows a simplified block diagram. It does not reflect the different input modes (especially the differential case), but it matches both single-ended cases.



SENSe2:FREQuency:CDR:RANGe?

IVI-COM Equivalent IAgilentN490xEDCDR.GetRanges (not IVI-compliant)

Syntax SENSe2:FREQuency:CDR:RANGe?

Description Comma-separated list of CDR (Hertz) ranges:
“min,max,min,max,min,max”

For example:
2.45e+009,3.21e+009,4.9e+009,6.42e+009,9.9e+009,1.09e+010

This indicates the following CDR ranges:

- 2.45 – 3.21 GHz
- 4.9 – 6.42 GHz
- 9.9 – 10.9 GHz

SENSe2:FREQuency:CDR:AUTo[?]

IVI-COM Equivalent IAgilentN490xEDCDR.ThresholdAutoTracking (not IVI-compliant)

Syntax SENSe2:FREQuency:CDR:AUTo 0|1|OFF|ON

SENSe2:FREQuency:CDR:AUT0?

Description Enables/disables the automatic tracking for the CDR threshold. The query returns the current setting. Default is ON.

The automatic tracking should be disabled for burst applications.

SENSe2:FREQuency:CDR:THReshold[:VALue][?]

IVI-COM Equivalent IAgilentN490xEDCDR.Threshold (not IVI-compliant)

Syntax SENSe2:FREQuency:CDR:THReshold:VALue <NR3>

SENSe2:FREQuency:CDR:THReshold:VALue?

Description Sets/gets the manual threshold for CDR bit transitions.

SENSe2:FREQuency:CDR:THReshold:MEASure[?]

IVI-COM Equivalent IAgilentN490xEDCDR.MeasureAndSetThreshold (not IVI-compliant)

Syntax SENSe2:FREQuency:CDR:THReshold:MEASure ONCe

SENSe2:FREQuency:CDR:THReshold:MEASure?

Description The command measures the DC level at the CDR input and sets the measured value as CDR threshold.

The query returns the current DC level at the CDR input.

SENSe2:VOLTage:EDGe[?]

IVI-COM Equivalent IAgilentN490xEDClockIn.ActiveEdge (IVI-compliant)

Syntax SENSe2:VOLTage:EDGe NEGative | POSitive

SENSe2:VOLTage:EDGe?

Description Sets the active edge of the clock input:

- NEGative

  The falling edge starts the period in which the input data is sampled.

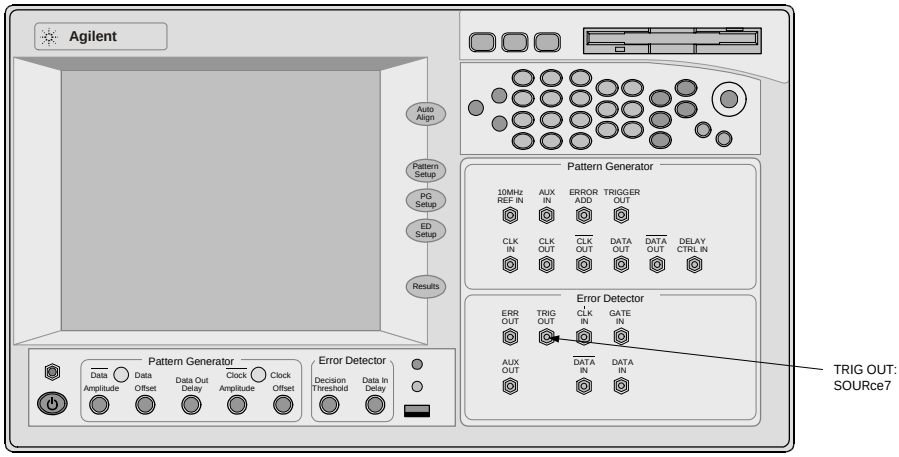
- POSitive

  The rising edge starts the period in which the input data is sampled.

This command has restrictions for frequencies under 620 Mbits/s. See the Serial BERT User Guide (or online Help) for details.

SOURce7 Subsystem

SOURce7 represents the error detector's Trigger Out port.



This subsystem has the following SCPI structure:

```
SOURce7
└ :TRIGger
  ├ [:MODE][?]
  ├ DCDRatio
  └ CTDRatio?
```

SOURce7:TRIGger[:MODe][?]

IVI-COM Equivalent IAgilentN490xEDTrigger.Mode (IVI-compliant)

Syntax SOURce7:TRIGger[:MODE] DCLock | PATTern

SOURce7:TRIGger[:MODE]?

Description The command configures the error detector's Trigger Out port from the error detector to be either:

- DCLock

  Divided clock mode (a square wave at clock rate/8)

- PATTern

  Pattern mode (a pulse synchronized to repetitions of the pattern)

The query returns current mode for the Trigger Out of the error detector.

SOURce7:TRIGger:DCDRatio

IVI-COM Equivalent IAgilentN490xEDTrigger.ClockDivisionRate (IVI-compliant)

Syntax SOURce7:TRIGger:DCDRatio <NR1>

Description Sets the trigger subratio.

SOURce7:TRIGger:CTDRatio?

IVI-COM Equivalent IAgilentN490xEDTrigger.ClockDivisionRate (IVI-compliant)

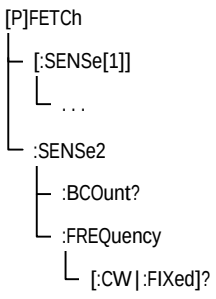
Syntax SOURce7:TRIGger:CTDratio?

Description Returns the trigger subratio.

[P]FETCh Subsystem

The [P]FETCh Subsystem is used to query the error detector's results. The PFETch subsystem returns the results immediately previously to the current results.

This subsystem has the following SCPI structure:



This subsystem has the following commands:

Name	Description under
Commands	
SENSe2:BCOunt?	*"[P]FETCh:SENSe2:BCOunt?" on page 156*
SENSe2:FREQ[:CW|:FIXed]?	*"[P]FETCh:SENSe2:FREQuency[:CW|:FIXed]?" on page 156*
Subnode	
[SENSe[1]]	*"[P]FETCh[:SENSe[1]] Subnode" on page 157*

[P]FETCh:SENSe2:BCOunt?

IVI-COM Equivalent IAgilentN490xEDMeasurement.ReadBitCount (IVI-compliant)

Syntax [P]FETCh:SENSe2:BCOunt?

Description This query returns the accumulated bit count since the start of the accumulation period.

[P]FETCh:SENSe2:FREQuency[:CW|:FIXed]?

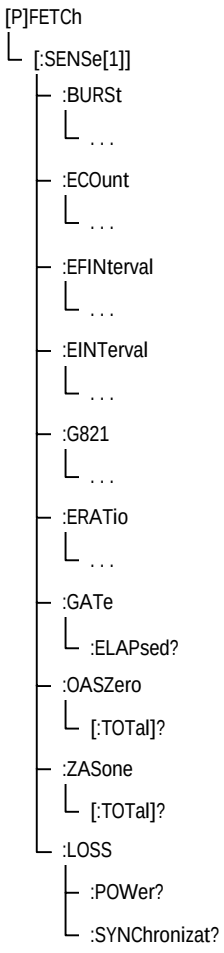
IVI-COM Equivalent IAgilentN490xEDMeasurement.ReadClockFrequency (IVI-compliant)

Syntax [P]FETCh:SENSe2:FREQuency[:CW|:FIXed]?

Description This query returns the current frequency of the signal on the clock input. This measurement is independent of the accumulation period.

[P]FETCh[:SENSe[1]] Subnode

This subnode has the following SCPI structure:



This subnode has the following commands and subnodes:

Name	Description under
Commands	
:GATe:ELAPsed?	*"[P]FETCh[:SENSe[1]]:GATe:ELAPsed?" on page 158*
:LOSS:POWer?	*"[P]FETCh[:SENSe[1]]:LOSS:POWer?" on page 159*
:LOSS:SYNChronizat?	*"[P]FETCh[:SENSe[1]]:LOSS:SYNChronizat?" on page 159*
Subnodes	
:BURSt	*"[P]FETCh[:SENSe[1]]:BURSt Subnode" on page 159*
:ECOunt	*"[P]FETCh[:SENSe[1]]:ECOunt Subnode" on page 162*
:EFINterval	*"[P]FETCh[:SENSe[1]]:EFINterval Subnode" on page 163*
:EINTerval	*"[P]FETCh[:SENSe[1]]:EINTerval Subnode" on page 165*
:ERATio	*"[P]FETCh[:SENSe[1]]:ERATio Subnode" on page 166*

[P]FETCh[:SENSe[1]]:GATe:ELAPsed?

IVI-COM Equivalent IAgilentN490xEDAccumulation.FetchElapsed (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:GATe:ELAPsed?

Description This query returns information about the degree to which the accumulation period has progressed.

If SENSe[1]:GATE:MANNer TIME is selected, then this command returns the elapsed time in the accumulation period in units of seconds.

If SENSe[1]:GATE:MANNer ERRors is selected, this command returns the elapsed errors into the accumulation period.

If SENSe[1]:GATE:MANNer BITS is selected, then this command returns the elapsed clock bits into the accumulation period.

[P]FETCh[:SENSe[1]]:LOSS:POWer?

Syntax [P]FETCh[:SENSe[1]]:LOSS:POWer?

Description This query returns the total number of seconds that power was lost since the start of the accumulation period.

This command is not supported.

[P]FETCh[:SENSe[1]]:LOSS:SYNChronizat?

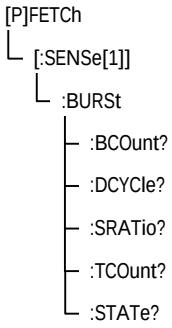
IVI-COM Equivalent IAgilentN490xEDMeasurement.ReadSecondsSyncLoss (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:LOSS:SYNChronizat?

Description This query returns of the total number of seconds for which the incoming pattern was not synchronized to the reference pattern during the accumulation period.

[P]FETCh[:SENSe[1]]:BURSt Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:BCOunt?	*"[P]FETCh[:SENSe[1]]:BURSt:BCOunt?" on page 160*
:DCYCle?	*"[P]FETCh[:SENSe[1]]:BURSt:DCYCle?" on page 160*
:SRATio?	*"[P]FETCh[:SENSe[1]]:BURSt:SRATio?" on page 160*
:TCOunt?	*"[P]FETCh[:SENSe[1]]:BURSt:TCOunt?" on page 160*
:STATe?	*"[P]FETCh[:SENSe[1]]:BURSt:STATe?" on page 161*

[P]FETCh[:SENSe[1]]:BURSt:BCOunt?

IVI-COM Equivalent IAgilentN490xEDBurst.ReadBadCount (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:BURSt:BCOunt?

Description This query returns the **B**ad Burst **CO**unt since the start of the accumulation period. If Burst mode is OFF, it returns 9.91E+37 (Not-A-Number, NAN).

For more information on how the Serial BERT handles burst measurements, see the Serial BERT User Guide (or online Help).

[P]FETCh[:SENSe[1]]:BURSt:DCYCle?

IVI-COM Equivalent IAgilentN490xEDBurst.ReadDutyCycle (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:BURSt:DCYCle?

Description This query is obsolete. It returns 9.91E+37 (Not-A-Number, NAN).

[P]FETCh[:SENSe[1]]:BURSt:SRATio?

IVI-COM Equivalent IAgilentN490xEDBurst.ReadSyncRatio (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:BURSt:SRATio?

Description This query returns the Burst **S**ynchronization **RAT**io since the start of the accumulation period. If Burst mode is OFF, it returns 9.91E+37 (Not-A-Number, NAN).

[P]FETCh[:SENSe[1]]:BURSt:TCOunt?

IVI-COM Equivalent IAgilentN490xEDBurst.ReadTotalCount (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:BURSt:TCOunt?

Description This query returns the **T**otal Burst **CO**unt since the start of the accumulation period. If Burst mode is OFF, it returns 9.91E+37 (Not-A-Number, NAN).

[P]FETCh[:SENSe[1]]:BURSt:STATe?

IVI-COM Equivalent IAgilentN490xEDBurst.ReadState (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:BURSt:STATe?

Description This query returns the burst state. While the accumulation period is running, the burst state always indicates, for example, if the current gate is too long or not. Thus, if the duty cycle or frequency of the gate signal is changed, the burst state also changes.

It may be of interest to know if an errored state occured during a measurement. For this reason, any errored state is stored even if the warning was only active for one burst out of thousands.

The following states may be returned:

- `BURST_RESULT_STATE__NO_ERROR`

  No burst errors have occurred (any of those listed below).

- `BURST_RESULT_STATE__GATE_SIGNAL_TOO_LONG`

  The Gate In signal is too long. This error can occur if there are too many bits within a burst. The limit is 4 Gbit. The length of the Gate In signal therefore depends on the bit rate.

  At 13 Gb/s, this state occurs roughly after 0.3 s (at slower bit rates, this occurs later). This error has a higher priority than “no unique 48bits”.

- `BURST_RESULT_STATE__NO_UNIQUE_48BITS_FOUND`

  For reliable synchronization, a pattern must contain a unique 48-bit pattern (the detect word). If the current pattern does not have a detect word, this error occurs.

  If this status occurs, the synchronization may be incorrect, as could also the measured bit error rate. There are standard patterns that may contain more than one instance of the used detect word. Statistically, every other burst would be correctly synchronized.

  In this case, it is recommended that you redefine the pattern. This error can only occur with memory-based patterns.

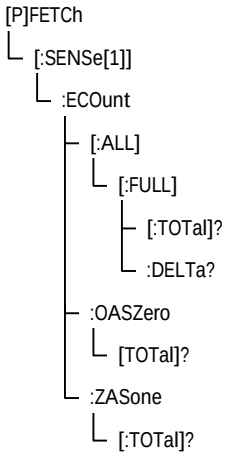
- `BURST_RESULT_STATE__UNKNOWN`

  The status is unknown. This can occur if accumulation has not been started, or if Burst Sync mode has not been activated.

NOTE Please note that the responses each have two underscores after STATE.

[P]FETCh[:SENSe[1]]:ECOunt Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:ECOunt[:ALL][:FULL][:TOTal]?	*"[P]FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL][:TOTal]?" on page 162*
:ECOunt[:ALL][:FULL]:DELTa?	*"[P]FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL]:DELTa?" on page 163*
:ECOunt:OASZero[:TOTal]?	*"[P]FETCh[:SENSe[1]]:ECOunt:OASZero[:TOTal]?" on page 163*
:ECOunt:ZASone[:TOTal]?	*"[P]FETCh[:SENSe[1]]:ECOunt:ZASone[:TOTal]?" on page 163*

[P]FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL][:TOTal]?

IVI-COM Equivalent IAgilentN490xEDErrorCount.Read (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL][:TOTal]?

Description This query is a contraction of the phrase **E**rror **COU**nt. It is the number of errors received in a time interval.

[P]FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL]:DELTa?

IVI-COM Equivalent IAgilentN490xEDErrorCount.ReadDelta (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL]:DELTa?

Description The “instantaneous” number of errors, calculated from the counts obtained in the last decisecond. This value is available even when accumulation is turned off.

[P]FETCh[:SENSe[1]]:ECOunt:OASZero[:TOTal]?

IVI-COM Equivalent IAgilentN490xEDErrorCount.ReadOAZ (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:ECOunt:OASZero[:TOTal]?

Description This is a contraction of the phrase **O**ne received **AS Z**ero. The command returns the number of errored ones (each true data one that is received as a data zero).

[P]FETCh[:SENSe[1]]:ECOunt:ZASone[:TOTal]?

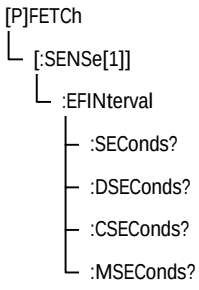
IVI-COM Equivalent IAgilentN490xEDErrorCount.ReadZAO (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:ECOunt:ZASone[:TOTal]?

Description This is a contraction of the phrase **Z**ero received **AS** one. The command returns the number of errored zeros (each true data zero that is received as a data one).

[P]FETCh[:SENSe[1]]:EFINterval Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:SEConds?	*"[P]FETCh[:SENSe[1]]:EFINterval:SEConds?" on page 164*
:DSEConds?	*"[P]FETCh[:SENSe[1]]:EFINterval:DSEConds?" on page 164*
:CSEConds?	*"[P]FETCh[:SENSe[1]]:EFINterval:CSEConds?" on page 164*
:MSEConds?	*"[P]FETCh[:SENSe[1]]:EFINterval:MSEConds?" on page 165*

[P]FETCh[:SENSe[1]]:EFINterval:SEConds?

IVI-COM Equivalent IAgilentN490xEDIntervals.ReadErrorFreeSeconds (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:EFINterval:SEConds?

Description This query is a contraction of the phrase **E**rror-**F**ree **IN**terval and returns a count of the number of seconds during which no error was detected.

[P]FETCh[:SENSe[1]]:EFINterval:DSEConds?

IVI-COM Equivalent IAgilentN490xEDIntervals.ReadErrorFreeDeciSeconds (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:EFINterval:DSEConds?

Description This query is a contraction of the phrase **E**rror-**F**ree **IN**terval and returns a count of the number of deciseconds during which no error was detected.

[P]FETCh[:SENSe[1]]:EFINterval:CSEConds?

IVI-COM Equivalent IAgilentN490xEDIntervals.ReadErrorFreeCentiSeconds (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:EFINterval:CSEConds?

Description This query is a contraction of the phrase **E**rror-**F**ree **IN**terval and returns a count of the number of centiseconds during which no error was detected.

[P]FETCh[:SENSe[1]]:EFINterval:MSEConds?

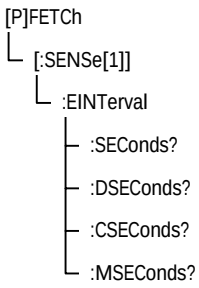
IVI-COM Equivalent IAgilentN490xEDIntervals.ReadErrorFreeMilliSeconds (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:EFINterval:MSEConds?

Description This query is a contraction of the phrase **E**rror-**F**ree **IN**terval and returns a count of the number of milliseconds during which no error was detected.

[P]FETCh[:SENSe[1]]:EINTerval Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:SEConds?	*"[P]FETCh[:SENSe[1]]:EINTerval:SEConds?" on page 165*
:DSEConds?	*"[P]FETCh[:SENSe[1]]:EINTerval:DSEConds?" on page 166*
:CSEConds?	*"[P]FETCh[:SENSe[1]]:EINTerval:CSEConds?" on page 166*
:MSEConds?	*"[P]FETCh[:SENSe[1]]:EINTerval:MSEConds?" on page 166*

[P]FETCh[:SENSe[1]]:EINTerval:SEConds?

IVI-COM Equivalent IAgilentN490xEDIntervals.ReadErroredSeconds (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:EINTerval:SEConds?

Description This query is a contraction of the phrase **E**rrored **INT**erval and returns a count of the number of seconds during which one or more errors were detected.

[P]FETCh[:SENSe[1]]:EINTerval:DSEConds?

IVI-COM Equivalent IAgilentN490xEDIntervals.ReadErroredDeciSeconds (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:EINTerval:DSEConds?

Description This query is a contraction of the phrase **E**rrored **INT**erval and returns a count of the number of deciseconds during which one or more errors were detected.

[P]FETCh[:SENSe[1]]:EINTerval:CSEConds?

IVI-COM Equivalent IAgilentN490xEDIntervals.ReadErroredCentiSeconds (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:EINTerval:CSEConds?

Description This query is a contraction of the phrase **E**rrored **INT**erval and returns a count of the number of centiseconds during which one or more errors were detected.

[P]FETCh[:SENSe[1]]:EINTerval:MSEConds?

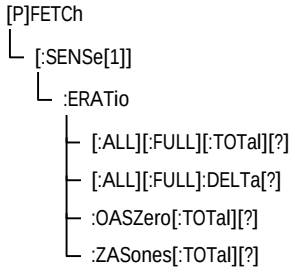
IVI-COM Equivalent IAgilentN490xEDIntervals.ReadErroredMilliSeconds (not IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:EINTerval:MSEConds?

Description This query is a contraction of the phrase **E**rrored **INT**erval and returns a count of the number of milliseconds during which one or more errors were detected.

[P]FETCh[:SENSe[1]]:ERATio Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
[:ALL][:FULL][:TOTal][?]	*"[P]FETCh[:SENSe[1]]:ERATio[:ALL][:FULL][:TOTal]?" on page 167*
[:ALL][:FULL]:DELTa[?]	*"[P]FETCh[:SENSe[1]]:ERATio[:ALL][:FULL]:DELTa?" on page 167*
:OASZero[:TOTal][?]	*"[P]FETCh[:SENSe[1]]:ERATio:OASZero[:TOTal]?" on page 167*
:ZASone[:TOTal][?]	*"[P]FETCh[:SENSe[1]]:ERATio:ZASone[:TOTal]?" on page 168*

[P]FETCh[:SENSe[1]]:ERATio[:ALL][:FULL][:TOTal]?

IVI-COM Equivalent IAgilentN490xEDErrorRatio.Read (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:ERATio[:ALL][:FULL][:TOTal]?

Description This query is a contraction of the phrase **E**rror **RAT**io. It is the ratio of the number of errors to the number of bits received in the interval specified by gate period.

[P]FETCh[:SENSe[1]]:ERATio[:ALL][:FULL]:DELTa?

IVI-COM Equivalent IAgilentN490xEDErrorRatio.ReadDelta (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:ERATio[:ALL][:FULL]:DELTa?

Description This query returns the "instantaneous" error ratio calculated from the counts obtained in the last decisecond. This value is available even when accumulation is turned off.

[P]FETCh[:SENSe[1]]:ERATio:OASZero[:TOTal]?

IVI-COM Equivalent IAgilentN490xEDErrorRatio.ReadOAZ (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:ERATio:OASZero[:TOTal]?

Description This is a contraction of the phrase **O**ne received **AS Z**ero. The query returns the ratio of erred ones (a true data one received a data zero) to number of bits.

[P]FETCh[:SENSe[1]]:ERATio:ZASone[:TOTal]?

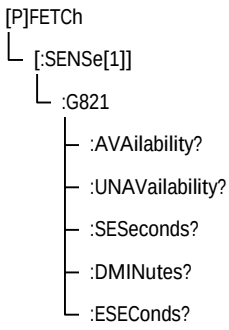
IVI-COM Equivalent IAgilentN490xEDErrorRatio.ReadZAO (IVI-compliant)

Syntax [P]FETCh[:SENSe[1]]:ERATio:ZASone[:TOTal]?

Description This is a contraction of the phrase **Z**ero received **AS O**ne. The query returns the ratio of erred zeros (a true data zero received a data one) to number of bits.

[P]FETCh[:SENSe[1]]:G821 Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:AVAilability?	*"[P]FETCh[:SENSe[1]]:G821:AVAilability?" on page 169*
:UNAVailability?	*"[P]FETCh[:SENSe[1]]:G821:UNAVailability?" on page 169*
:SESeconds?	*"[P]FETCh[:SENSe[1]]:G821:SESeconds?" on page 169*
:DMINutes?	*"[P]FETCh[:SENSe[1]]:G821:DMINutes?" on page 169*
:ESEConds?	*"[P]FETCh[:SENSe[1]]:G821:ESEConds?" on page 169*

NOTE The following commands return a percentage of seconds that have been classified according to the CCITT's G.821 specification.

[P]FETCh[:SENSe[1]]:G821:AVAilability?

Syntax [P]FETCh[:SENSe[1]]:G821:AVAilability?

Description Returns the G.821 *availability*.

[P]FETCh[:SENSe[1]]:G821:UNAVailability?

Syntax [P]FETCh[:SENSe[1]]:G821:UNAVailability?

Description Returns the G.821 *unavailability*.

[P]FETCh[:SENSe[1]]:G821:SESeconds?

Syntax [P]FETCh[:SENSe[1]]:G821:SESeconds?

Description Returns the G.821 *severely errored seconds*.

[P]FETCh[:SENSe[1]]:G821:DMINutes?

Syntax [P]FETCh[:SENSe[1]]:G821:DMINutes?

Description Returns the G.821 *degraded minutes*.

[P]FETCh[:SENSe[1]]:G821:ESEConds?

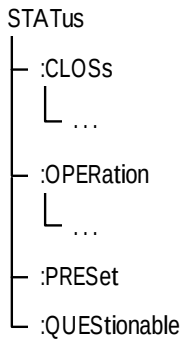
Syntax [P]FETCh[:SENSe[1]]:G821:ESEConds?

Description Returns the G.821 *errored seconds*.

STATus Subsystem

The STATus Subsystem provides an interface to the instrument's Status Register. For information on how to work with the Status register, see *"Serial BERT Register Model" on page 25*.

This subsystem has the following SCPI structure:



This subsystem has the following commands and subnodes:

Name	Description under
Commands	
STATus:PRESet	*"STATus:PRESet" on page 170*
Subnodes	
STATus:CLOSs	*"CLOSs Subnode" on page 171*
STATus:OPERation	*"STATus:OPERation Subnode" on page 173*
STATus:QUEStionable	*"STATus:QUEStionable Subnode" on page 176*

STATus:PRESet

IVI-COM Equivalent IAgilentN490xStatus.Preset (not IVI-compliant)

Syntax STATus:PRESet

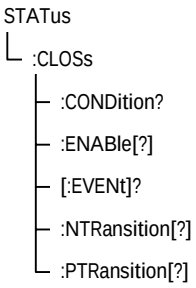
Description The PRESet command is an event that configures the SCPI and device-dependent status data structures, such that the device-dependent events are reported at a higher level through the mandatory part of the status reporting structures.

The PRESet command affects only the enable register and the transition filter registers for the SCPI mandated and device dependent status data structures. PRESet does not affect either the “status byte” or the “standard event status” as defined by IEEE 488.2. PRESet does not clear any of the event registers. The *CLS command is used to clear all event registers in the device status reporting mechanism.

From the device-dependent status data structures, the PRESet command sets the enable register to all one’s and the transition filter to recognize both positive and negative transitions. For the SCPI mandatory status data structures, the PRESet command sets the transition filter registers to recognize only positive transitions and sets the enable register to zero.

CLOSs Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:CONDition	*“STATus:CLOSs:CONDition” on page 171*
:ENABle[?]	*“STATus:CLOSs:ENABle[?]” on page 172*
[:EVENt]?	*“STATus:CLOSs[:EVENt]?” on page 172*
:NTRansition[?]	*“STATus:CLOSs:NTRansition[?]” on page 172*
:PTRansition[?]	*“STATus:CLOSs:PTRansition[?]” on page 173*

STATus:CLOSs:CONDition

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:CLOSs:CONDition?

Description This query returns the contents of the condition register in the Clock Loss Status Register. See *“Clock Loss Register” on page 28* for the layout of the Clock Loss register.

STATus:CLOSs:ENABle[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:CLOSs:ENABle <Num.>

STATus:CLOSs:ENABle?

Description The command sets the enable mask in the Clock Loss Register, which allows true conditions in the event register to be reported in the summary bit. The query returns the weighted value of the bits that are set in the enable register. See *"Clock Loss Register" on page 28* for the layout of the Clock Loss register.

STATus:CLOSs[:EVENt]?

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:CLOSs[:EVENt]?

Description The bits in this register indicate pattern generator and error detector clock loss. This query returns whether the pattern generator or error detector has experienced the clock loss. See *"Clock Loss Register" on page 28* for the layout of the Clock Loss register.

STATus:CLOSs:NTRansition[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:CLOSs:NTRansition <Num.>

STATus:CLOSs:NTRansition?

Description This command sets the negative transition register state in the Clock Loss Register. When a bit in this mask is set to "1", negative (logic 1 changing to logic 0) transitions of this bit are allowed to pass. The query returns the weighted value of the bits that are set to pass negative transitions in the transition filter. See *"Clock Loss Register" on page 28* for the layout of the Clock Loss register.

STATus:CLOSs:PTRansition[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

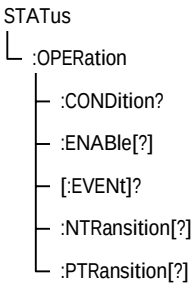
Syntax STATus:CLOSs:PTRansition <Num.>

STATus:CLOSs:PTRansition?

Description This command sets the positive transition register state in the Clock Loss Register. When a bit in this mask is set to "1", positive transitions (logic 0 changing to logic 1) of this bit are allowed to pass. This is the default setting of the instrument. The query returns the weighted value of the bits that are set to pass positive transitions in the transition filter. See *"Clock Loss Register" on page 28* for the layout of the Clock Loss register.

STATus:OPERation Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:CONDition?	*"STATus:OPERation:CONDition?" on page 174*
:ENABle[?]	*"STATus:OPERation:ENABle[?]" on page 174*
[:EVENt]?	*"STATus:OPERation[:EVENt]?" on page 174*
:NTRansition[?]	*"STATus:OPERation:NTRansition[?]" on page 175*
:PTRansition[?]	*"STATus:OPERation:PTRansition[?]" on page 175*

STATus:OPERation:CONDition?

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:OPERation:CONDition?

Description This query only returns the contents of the condition register in the Operation Status Register. See *"Operation Status Register" on page 30* for the layout of the Operation Status register.

STATus:OPERation:ENABle[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:OPERation:ENABle

STATus:OPERation:ENABle?

Description The command sets the enable mask in the Operation Status Register, which allows true conditions in the event register to be reported in the summary bit. The query returns the weighted value of the bits that are set in the enable register. See *"Operation Status Register" on page 30* for the layout of the Operation Status register.

STATus:OPERation[:EVENt]?

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:OPERation[:EVENt]?

Description This query returns the contents of the Operation Status event register. See *"Operation Status Register" on page 30* for the layout of the Operation Status register.

STATus:OPERation:NTRansition[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:OPERation:NTRansition

STATus:OPERation:NTRansition?

Description This command sets the transition filter state in the Operation Status Register. When this mask is set to "1", negative (logic 1 changing to logic 0) transitions are allowed to pass. The query returns the weighted value of the bits that are set to pass negative transitions in the transition filter. See *"Operation Status Register" on page 30* for the layout of the Operation Status register.

STATus:OPERation:PTRansition[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

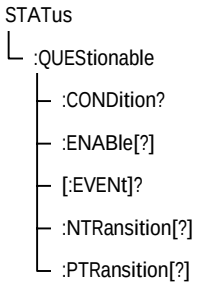
Syntax STATus:OPERation:PTRansition

STATus:OPERation:PTRansition?

Description This command sets the transition filter state in the Operation Status Register. When this mask is set to "1", positive transitions (logic 0 changing to logic 1) are allowed to pass. This is the default setting of the instrument. The query returns the weighted value of the bits that are set to pass positive transitions in the transition filter. See *"Operation Status Register" on page 30* for the layout of the Operation Status register.

STATus:QUEStionable Subnode

This subnode has the following SCPI structure:



This subnode has the following commands:

Name	Description under
:CONDition?	*"STATus:QUEStionable:CONDition?" on page 176*
:ENABle[?]	*"STATus:QUEStionable:ENABle[?]" on page 177*
[:EVENt]?	*"STATus:QUEStionable[:EVENt]?" on page 177*
:NTRansition[?]	*"STATus:QUEStionable:NTRansition[?]" on page 177*
:PTRansition[?]	*"STATus:QUEStionable:PTRansition[?]" on page 178*

STATus:QUEStionable:CONDition?

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:QUEStionable:CONDition?

Description This query returns the contents of the condition register in the Questionable Status Register. See *"Questionable Status Register" on page 29* for the layout of the Questionable Status register.

STATus:QUEStionable:ENABle[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:QUEStionable:ENABle

STATus:QUEStionable:ENABle?

Description The command form sets the enable mask in the Questionable Status Register, which allows true conditions in the event register to be reported in the summary bit. The query form returns the weighted value of the bits that are set in the enable register. See *"Questionable Status Register" on page 29* for the layout of the Questionable Status register.

STATus:QUEStionable[:EVENt]?

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:QUEStionable[:EVENt]

STATus:QUEStionable:ENABle?

Description This query form returns the contents of the Questionable Status event register. See *"Questionable Status Register" on page 29* for the layout of the Questionable Status register.

STATus:QUEStionable:NTRansition[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:QUEStionable:NTRansition

STATus:QUEStionable:NTRansition?

Description This command sets the transition filter state in the Questionable Status Register. When this mask is set to "1", negative (logic 1 changing to logic 0) transitions are allowed to pass. The query form returns the weighted value of the bits that are set to pass negative transitions in the transition filter. See*"Questionable Status Register" on page 29* for the layout of the Questionable Status register.

STATus:QUEStionable:PTRansition[?]

IVI-COM Equivalent IAgilentN490xStatus.Register (not IVI-compliant)

Syntax STATus:QUEStionable:PTRansition

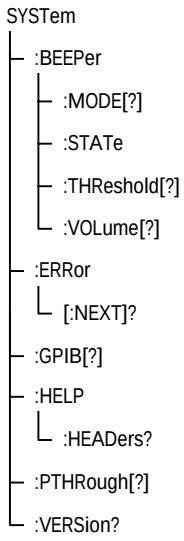
STATus:QUEStionable:PTRansition?

Description This command sets the transition filter state in the Questionable Status Register. When this mask is set to "1", positive transitions (logic 0 changing to logic 1) are allowed to pass. This is the default setting of the instrument. The query returns the weighted value of the bits that are set to pass positive transitions in the transition filter. See *"Questionable Status Register" on page 29* for the layout of the Questionable Status register.

SYSTem Subsystem

The SYSTem subsystem represents general system functions.

The subsystem has the following SCPI structure:



This subsystem has the following commands:

Name	Description under
:BEEPer:MODe[?]	*"SYSTem:BEEPer:MODe[?]" on page 179*
:BEEPer:STATe	*"SYSTem:BEEPer:STATe[?]" on page 179*
:BEEPer:THReshold[?]	*"SYSTem:BEEPer:THReshold[?]" on page 180*
:BEEPer:VOLume[?]	*"SYSTem:BEEPer:VOLume[?]" on page 180*
:ERRor[:NEXT]?	*"SYSTem:ERRor[:NEXT]?" on page 180*
:GPIB[?]	
:HELP:HEADers?	*"SYSTem:HELP:HEADers?" on page 181*
:VERSion?	*"SYSTem:VERSion?" on page 181*

SYSTem:BEEPer:MODe[?]

IVI-COM Equivalent IAgilentN490xAudio.Mode (not IVI-compliant)

Syntax SYSTem:BEEPer:MODE BERalarm | TONes

SYSTem:BEEPer:MODE?

Description The command form sets the instrument's audible beeper to trigger on either a specific BER level (BERalarm) or on any occurrence of errors (TONes).

The response returns the current mode setting of the instrument's audible beeper.

SYSTem:BEEPer:STATe[?]

IVI-COM Equivalent IAgilentN490xAudio.Enabled (not IVI-compliant)

Syntax SYSTem:BEEPer:STATe 0 | 1 | OFF | ON

SYSTem:BEEPer:STATe?

Description The command turns on and off the instrument's audible beeper. The response returns whether or not the instrument's audible beeper is turned on.

SYSTem:BEEPer:THReshold[?]

IVI-COM Equivalent IAgilentN490xAudio.Threshold (not IVI-compliant)

Syntax SYSTem:BEEPer:THReshold <Numeric value>

SYSTem:BEEPer:THREshold?

Description The command sets the BER threshold value at which the instrument's audible beeper will produce sounds.

The response returns the current setting of the BER threshold at which the instrument's audible beeper will produce sounds.

SYSTem:BEEPer:VOLume[?]

IVI-COM Equivalent IAgilentN490xAudio.Volume (not IVI-compliant)

Syntax SYSTem:BEEPer:VOLume <Numeric value>

SYSTem:BEEPer:VOLume?

The command controls the volume of the instrument's audible beeper. The response returns the current volume of the instrument's audible beeper.

SYSTem:ERRor[:NEXT]?

IVI-COM Equivalent IIviDriverOperation.GetNextInterchangeWarning (IVI-compliant)

Syntax SYSTem:ERRor[:NEXT]?

Description This query pulls the next error from the error queue, and returns the error number and a string describing the error. The error queue has a depth of 20.

SYSTem:GPIB[?]

IVI-COM Equivalent IAgilentN490xUtilities.GPIBAddress (not IVI-compliant)

Syntax SYSTem:GPIB <Numeric value>

SYSTem:GPIB?

Description Sets or returns the instrument's GPIB address.

SYSTem:HELP:HEADers?

Syntax SYSTem:HELP:HEADers?

Description This query returns the complete list of instrument commands. Not all of the commands are implemented, however. For more information, refer to the specific command groups in this guide.

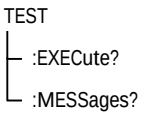
SYSTem:VERSion?

Syntax SYSTem:VERSion?

Description This query returns the version of the SCPI programming language, which supports the GPIB commands.

TEST Subsystem

The TEST Subsystem represents the instrument's selftest functions.



TEST:EXECute?

IVI-COM Equivalent IIviDriverUtility.SelfTest (IVI-compliant)

Syntax TEST:EXECute? [SelfTest_value] {,<SelfTest_value>}

Description This command runs user-specified self tests. If no parameter is specifed, all tests are run.

Successful completion of a self test returns 0. If a self test fails, 1 is returned.

SelfTest_value can be one of the parameters listed below.

Parameter	Description
ALL	Error detector and pulse generator module self test is started.
PGENerator	Pulse generator module self test is started.
EDETector	Error detector module self test is started.
PGCal	Auto calibration of pulse generator delay.
EDCal	Auto calibration of error detector delay.

NOTE Use TEST:MESS? to read the result of the self tests.

TEST:MESSages?

IVI-COM Equivalent IIviDriverUtility.ErrorQuery (IVI-compliant)

Syntax TEST:MESSages? PGPOn | EDPOn | EDET | PGEN

Description Returns a comma-separated list of messages. This command has the following options:

- PGPOn: Pattern Generator Power On messages
- EDPOn: Error Detector Power On messages
- PGEN: Pattern Generator selftest messages
- EDET: Error Detector selftest messages
- PGCal: Pattern Generator Calibration Results
- EDCal: Error Detector Calibration Results.

Appendix

This chapter discusses the differences between the Serial BERT and its precursors (the Agilent 71612C and 86130A) from the programmer's point of view. The information here is intended to help the programmer who is porting test applications that have been developed for these other products.

General Differences

The Serial BERT differs from both the Agilent 71612C and 86130A in the following points:

Added features

The following features have been added:

- Delay Control Input port and subsystem
- Clock Data Recovery (CDR) mode
- Output blanking
- Additional trigger dividers for the pattern generator and error detector

Modified functionality

The following functions have been modified:

- Error messages are completely different.
- Logging is handled differently.
- Status registers have been reorganized.
- Selftests have been modified.
- Termination voltage can now be set variably.
- Additional logic families are supported.
- The frequency range has been extended in both directions.
- Patterns up to 32 Mbits long are possible.

Features no longer supported

The following features are no longer supported:

- Enabling and disabling of the output and input ports
- The MMEMory subsystem
- Clock In polarity for the error detector
- Stretched mode at the Error Out port
- Open box?

Differences Specific to the Agilent 86130A

Added features

The following features have been added:

- Bit Error Location; you can specify a single bit or an entire block for error analysis.
- Zero substitution for PRBN patterns
- Further subsets of deci-second error intervals (milli- and centi-seconds)
- G821 error interval measurements

Features no longer supported

The following features are no longer supported:

- Separate levels for the CLOCK/$\overline{\text{CLOCK}}$ and DATA/$\overline{\text{DATA}}$ outputs.
- Quick alignment is no longer supported.

Differences Specific to the Agilent 71612C

Added features

The following features have been added:

- Additional internal alternate pattern modes.
- User files are supported.

Index

Printed in Germany May 2004

5989-0386EN

Agilent Technologies